VLM

January 2, 2026

1 LLaVA-Med for Binary Medical VQA on Chest X-rays

Model B: Vision-Language Model Approach

This notebook implements LLaVA-Med (Large Language and Vision Assistant for BioMedicine) for binary Visual Question Answering on the VQA-RAD dataset, focusing on Chest X-rays with Yes/No questions.

We use Parameter-Efficient Fine-Tuning (PEFT) with LoRA to adapt the model to our specific task.

Requirements: Local GPU with ~12 GB VRAM (4-bit quantization)

1.1 1. Environment Setup

[1]:
```
# Local setup (no Colab / no git clone)
# Assumes a local LLaVA checkout at ./LLaVA

import importlib.util

required = [
    "torch",
    "transformers",
    "bitsandbytes",
    "peft",
    "sklearn",
    "tqdm",
    "seaborn",
    "PIL",
]
missing = [pkg for pkg in required if importlib.util.find_spec(pkg) is None]
if missing:
    print("Missing packages:", missing)
    print("Install locally before continuing, e.g.:")
    print("  pip install -r ./LLaVA/requirements.txt")
    print("  pip install bitsandbytes>=0.41.0 peft>=0.7.0 scikit-learn tqdm␣
 ↪seaborn")
```

[2]:
```
import os

# Hugging Face token for faster downloads (set in your environment)
hf_token = os.environ.get("HUGGINGFACE_HUB_TOKEN") or os.environ.get("HF_TOKEN")
if hf_token:
    os.environ["HUGGINGFACE_HUB_TOKEN"] = hf_token
    os.environ["HF_TOKEN"] = hf_token
    print("HF token detected for this session.")
else:
    print("No HF token found. Set HUGGINGFACE_HUB_TOKEN or HF_TOKEN for faster␣
 ↪downloads.")
```

```
No HF token found. Set HUGGINGFACE_HUB_TOKEN or HF_TOKEN for faster downloads.
```

[3]:
```
import os
import sys
from pathlib import Path

ROOT_DIR = Path.cwd()
LLAVA_DIR = ROOT_DIR / "LLaVA"
if not LLAVA_DIR.exists():
    raise FileNotFoundError(f"LLaVA repo not found at: {LLAVA_DIR}")

sys.path.insert(0, str(LLAVA_DIR))

import json
import random
import warnings
import numpy as np
import torch
from PIL import Image
from collections import defaultdict
from tqdm import tqdm
from sklearn.metrics import accuracy_score, f1_score, classification_report,␣
 ↪confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

warnings.filterwarnings("ignore")

# Check GPU + VRAM (target: ~12 GB)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Device: {device}")
if torch.cuda.is_available():
    vram_gb = torch.cuda.get_device_properties(0).total_memory / 1e9
    print(f"GPU: {torch.cuda.get_device_name(0)}")
    print(f"Memory: {vram_gb:.1f} GB")
```

```
    if vram_gb < 11.5:
        print("Warning: GPU has < 12 GB VRAM. You may need to reduce batch size␣
↪or max tokens.")
```

```
Device: cuda
GPU: NVIDIA GeForce RTX 4070 Ti
Memory: 12.9 GB
```

[4]:
```
# Reproducibility - SAME SEED as CNN baseline
SEED = 777
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed(SEED)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
```

1.2 2. Dataset Loading and Preprocessing

Using the **exact same filtering and splitting strategy** as the CNN baseline to ensure fair comparison.

The VQA_RAD dataset should be in the local `./VQA_RAD` folder.

[5]:
```
from pathlib import Path

# Dataset paths - VQA_RAD in the project folder
ROOT_DIR = Path.cwd()
DATA_DIR = ROOT_DIR / "VQA_RAD"
IMAGE_DIR = DATA_DIR / "VQA_RAD Image Folder"
JSON_PATH = DATA_DIR / "VQA_RAD Dataset Public.json"

# Verify dataset exists
if not JSON_PATH.exists():
    print("Dataset not found at expected path!")
    print(f"Expected: {JSON_PATH}")
    print("Please place the VQA_RAD folder here:")
    print(f"  {DATA_DIR}")
    print("  VQA_RAD/")
    print("    VQA_RAD Image Folder/")
    print("      synpic0001.jpg")
    print("      ...")
    print("    VQA_RAD Dataset Public.json")
else:
    print("Dataset found!")
    print(f"Images: {IMAGE_DIR}")
    print(f"JSON: {JSON_PATH}")
```

```
Dataset found!
```

```
Images: c:\Users\Yousef\Desktop\yurdurhurhud\VQA_RAD\VQA_RAD Image Folder
JSON: c:\Users\Yousef\Desktop\yurdurhurhud\VQA_RAD\VQA_RAD Dataset Public.json
```

[6]:
```python
# Load raw data
with open(JSON_PATH, "r") as f:
    raw_data = json.load(f)

print(f"Total raw samples: {len(raw_data)}")
```

```
Total raw samples: 2248
```

[7]:
```python
def normalize_answer(ans):
    """Normalize answer to binary label (same as CNN baseline)."""
    ans = str(ans).lower().strip()
    if ans in ["yes", "y"]:
        return 1
    if ans in ["no", "n"]:
        return 0
    return None

# Filter dataset: Chest X-rays only, Binary (Yes/No) questions only
# EXACT SAME FILTERING AS CNN BASELINE
samples = []

for item in raw_data:
    # Filter for Chest images only
    if item.get("image_organ", "").lower() != "chest":
        continue

    # Filter for closed-ended (binary) questions only
    if item.get("answer_type", "").lower() != "closed":
        continue

    # Normalize answer to binary
    label = normalize_answer(item.get("answer", ""))
    if label is None:
        continue

    image_name = item.get("image_name")
    if image_name is None:
        continue

    samples.append({
        "image_path": os.path.join(IMAGE_DIR, image_name),
        "image_id": image_name,
        "question": item["question"],
        "label": label,
        "answer_text": "yes" if label == 1 else "no"
```

```
    })

print(f"Filtered samples (Chest X-ray, Binary): {len(samples)}")

# Check class distribution
yes_count = sum(1 for s in samples if s["label"] == 1)
no_count = len(samples) - yes_count
print(f"Class distribution - Yes: {yes_count} ({yes_count/len(samples)*100:.
 ↪1f}%), No: {no_count} ({no_count/len(samples)*100:.1f}%)")
```

```
Filtered samples (Chest X-ray, Binary): 477
Class distribution - Yes: 186 (39.0%), No: 291 (61.0%)
```

[8]:
```
def image_level_split(samples, seed=SEED):
    """
    Image-level splitting to prevent data leakage.
    All questions for the same image go to the same split.
    EXACT SAME SPLITTING AS CNN BASELINE.
    """
    random.seed(seed)

    # Group samples by image
    image_to_samples = defaultdict(list)
    for s in samples:
        image_to_samples[s["image_id"]].append(s)

    # Shuffle image IDs
    image_ids = list(image_to_samples.keys())
    random.shuffle(image_ids)

    # Split: 80% train, 10% val, 10% test
    n = len(image_ids)
    train_ids = image_ids[:int(0.8 * n)]
    val_ids = image_ids[int(0.8 * n):int(0.9 * n)]
    test_ids = image_ids[int(0.9 * n):]

    def collect(ids):
        out = []
        for i in ids:
            out.extend(image_to_samples[i])
        return out

    return collect(train_ids), collect(val_ids), collect(test_ids)

train_samples, val_samples, test_samples = image_level_split(samples)
print(f"Train: {len(train_samples)}, Val: {len(val_samples)}, Test:␣
 ↪{len(test_samples)}")
```

```
Train: 365, Val: 49, Test: 63
```

[9]:
```
# Verify split matches CNN baseline (should be 365, 49, 63)
print(f"\nExpected from CNN baseline: Train=365, Val=49, Test=63")
print(f"Actual: Train={len(train_samples)}, Val={len(val_samples)},␣
 ↪Test={len(test_samples)}")

# Inspect some samples
print("\nSample questions:")
for i, s in enumerate(train_samples[:3]):
    print(f"  {i+1}. Q: {s['question']}")
    print(f"     A: {s['answer_text']}\n")
```

```
Expected from CNN baseline: Train=365, Val=49, Test=63
Actual: Train=365, Val=49, Test=63

Sample questions:
  1. Q: Is this in the PA plane?
     A: yes

  2. Q: Is this modality safe for pregnant women?
     A: no

  3. Q: Do you see a pleural effusion?
     A: no
```

1.3 3. Load LLaVA-Med Model

Using the official LLaVA codebase with 4-bit quantization to fit in ~12 GB VRAM.

[10]:
```
from llava.model.builder import load_pretrained_model
from llava.mm_utils import get_model_name_from_path, process_images,␣
 ↪tokenizer_image_token
from llava.constants import IMAGE_TOKEN_INDEX, DEFAULT_IMAGE_TOKEN,␣
 ↪DEFAULT_IM_START_TOKEN, DEFAULT_IM_END_TOKEN
from llava.conversation import conv_templates
```

[11]:
```
import os
from pathlib import Path
from transformers import BitsAndBytesConfig

MODEL_PATH = os.environ.get("LLAVA_MED_MODEL_PATH", "microsoft/llava-med-v1.
 ↪5-mistral-7b")
MAX_VRAM_GB = 12

if Path(MODEL_PATH).exists():
```

```
    print(f"Using local model path: {MODEL_PATH}")
else:
    print(f"Using Hugging Face model ID: {MODEL_PATH}")
    print("Tip: set LLAVA_MED_MODEL_PATH to a local folder to avoid downloads.")

print("Loading LLaVA-Med model (this may take a few minutes)...")

max_memory = None
if torch.cuda.is_available():
    max_memory = {i: f"{MAX_VRAM_GB}GB" for i in range(torch.cuda.
 ↪device_count())}

# Use HF token from env if present (avoids guest rate limits)
hf_token = os.environ.get("HUGGINGFACE_HUB_TOKEN") or os.environ.get("HF_TOKEN")

# Newer transformers disallow load_in_4bit with quantization_config.
# Use quantization_config only (see LLaVA issue #1638).
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_compute_dtype=torch.float16,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4"
)

MODEL_LOAD_KWARGS = dict(
    model_path=MODEL_PATH,
    model_base=None,
    model_name=get_model_name_from_path(MODEL_PATH),
    load_4bit=False,
    device_map="auto",
    quantization_config=bnb_config
)
if max_memory is not None:
    MODEL_LOAD_KWARGS["max_memory"] = max_memory
if hf_token:
    MODEL_LOAD_KWARGS["token"] = hf_token

tokenizer, model, image_processor, context_len =␣
 ↪load_pretrained_model(**MODEL_LOAD_KWARGS)

print("Model loaded successfully!")
print(f"Context length: {context_len}")
```

```
Using Hugging Face model ID: microsoft/llava-med-v1.5-mistral-7b
Tip: set LLAVA_MED_MODEL_PATH to a local folder to avoid downloads.
Loading LLaVA-Med model (this may take a few minutes)…
```

```
`torch_dtype` is deprecated! Use `dtype` instead!
```

```
Loading checkpoint shards:   0%|          | 0/4 [00:00<?, ?it/s]

Some weights of the model checkpoint at microsoft/llava-med-v1.5-mistral-7b were
not used when initializing LlavaMistralForCausalLM:
['model.vision_tower.vision_tower.vision_model.embeddings.class_embedding', 'mod
el.vision_tower.vision_tower.vision_model.embeddings.patch_embedding.weight', 'm
odel.vision_tower.vision_tower.vision_model.embeddings.position_embedding.weight
', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_norm1.bi
as', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_norm1.
weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_no
rm2.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_
norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.0.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.0.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.0.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.0.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.0.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.1.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.1.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.1.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.1.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.1.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.1.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.1.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.1.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.1.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.10.layer_norm1.bias', 'model.vi
sion_tower.vision_tower.vision_model.encoder.layers.10.layer_norm1.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.10.layer_norm2.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.10.layer_norm2.weigh
t',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc1.bias',
```

'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
0.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.10.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.10.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.10.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.10.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.10.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
1.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.11.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.11.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.11.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.11.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.11.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
2.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.12.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.12.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.12.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.12.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.12.self_attn.v_proj.weight', 'mod

el.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.13.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
3.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.13.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.13.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.13.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.13.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.13.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.14.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
4.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.14.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.14.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.14.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.14.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.14.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.15.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1

5.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.15.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.15.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.15.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.15.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.15.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.16.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
6.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.16.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.16.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.16.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.16.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.16.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.17.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
7.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.17.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.17.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.17.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.17.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.17.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc1.bias',

'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.18.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
8.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.18.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.18.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.18.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.18.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.18.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.19.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
9.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.19.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.19.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.19.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.19.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.19.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm1.bias', 'm
odel.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm1.weight'
, 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm2.bia
s', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm2.w
eight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.2.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.2.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.2.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.2.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.2.self_attn.v_proj.weight', 'model.visio

n_tower.vision_tower.vision_model.encoder.layers.20.layer_norm1.bias', 'model.vi
sion_tower.vision_tower.vision_model.encoder.layers.20.layer_norm1.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.20.layer_norm2.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.20.layer_norm2.weigh
t',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.20.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
0.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.20.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.20.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.20.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.20.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.20.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.21.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
1.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.21.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.21.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.21.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.21.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.21.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.22.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2

2.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.22.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.22.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.22.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.22.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.22.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.23.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
3.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.23.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.23.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.23.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.23.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.23.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm1.bias', 'm
odel.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm1.weight'
, 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm2.bia
s', 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm2.w
eight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.3.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.3.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.3.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.3.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.3.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.4.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.4.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.4.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.4.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc1.weight',

'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.4.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.4.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.4.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.4.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.4.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.4.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.4.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.5.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.5.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.5.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.5.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.5.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.5.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.5.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.5.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.5.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.5.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.5.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.6.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.6.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.6.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.6.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.6.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.6.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.6.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.6.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.6.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.6.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.6.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.7.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.7.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.7.layer_norm2.bias', 'mod

el.vision_tower.vision_tower.vision_model.encoder.layers.7.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.7.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.7.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.7.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.7.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.7.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.7.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.7.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.8.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.8.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.8.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.8.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.8.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.8.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.8.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.8.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.8.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.8.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.8.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.9.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.9.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.9.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.9.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.9.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.9.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.9.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.9.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.9.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.9.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.9.self_attn.v_proj.weight',

```
'model.vision_tower.vision_tower.vision_model.post_layernorm.bias',
'model.vision_tower.vision_tower.vision_model.post_layernorm.weight',
'model.vision_tower.vision_tower.vision_model.pre_layrnorm.bias',
'model.vision_tower.vision_tower.vision_model.pre_layrnorm.weight']
- This IS expected if you are initializing LlavaMistralForCausalLM from the
checkpoint of a model trained on another task or with another architecture (e.g.
initializing a BertForSequenceClassification model from a BertForPreTraining
model).
- This IS NOT expected if you are initializing LlavaMistralForCausalLM from the
checkpoint of a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).

Model loaded successfully!
Context length: 2048
```

[12]:
```python
# Check conversation template
CONV_MODE = "mistral_instruct"  # LLaVA-Med v1.5 uses Mistral
print(f"Conversation mode: {CONV_MODE}")

# Test conversation template
conv = conv_templates[CONV_MODE].copy()
print(f"System message: {conv.system}")
```

```
Conversation mode: mistral_instruct
System message:
```

1.4 4. Setup LoRA for Fine-Tuning

[13]:
```python
from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training,␣
 ↪TaskType

# Prepare model for k-bit training
model = prepare_model_for_kbit_training(model)

# LoRA configuration targeting the language model layers
lora_config = LoraConfig(
    r=16,                           # Rank of the update matrices
    lora_alpha=32,                  # Scaling factor
    lora_dropout=0.1,               # Dropout probability
    bias="none",
    task_type=TaskType.CAUSAL_LM,
    target_modules=[                # Target attention and MLP layers
        "q_proj",
        "k_proj",
        "v_proj",
        "o_proj",
        "gate_proj",
        "up_proj",
```

```
        "down_proj"
    ]
)

# Apply LoRA
model = get_peft_model(model, lora_config)
model.print_trainable_parameters()
```

```
trainable params: 44,302,336 || all params: 7,610,521,600 || trainable%: 0.5821
```

1.5 5. Inference Functions

[14]:
```
# System prompt for binary VQA
SYSTEM_PROMPT = """You are a medical assistant specialized in analyzing Chest␣
 ↪X-rays. Answer the following question about this Chest X-ray with ONLY 'yes'␣
 ↪or 'no'. Do not provide any explanation."""

def create_prompt(question):
    """Create the prompt for the model."""
    return f"{SYSTEM_PROMPT}\n\nQuestion: {question}\nAnswer:"

def prepare_inputs(image_path, question, tokenizer, image_processor, model):
    """
    Prepare inputs for the model.
    """
    # Load and process image
    image = Image.open(image_path).convert("RGB")
    image_tensor = process_images([image], image_processor, model.config)
    image_tensor = image_tensor.to(model.device, dtype=torch.float16)

    # Create conversation
    conv = conv_templates[CONV_MODE].copy()

    # Add image token and question
    prompt = create_prompt(question)
    inp = DEFAULT_IMAGE_TOKEN + "\n" + prompt

    conv.append_message(conv.roles[0], inp)
    conv.append_message(conv.roles[1], None)

    full_prompt = conv.get_prompt()

    # Tokenize
    input_ids = tokenizer_image_token(
        full_prompt,
        tokenizer,
        IMAGE_TOKEN_INDEX,
```

```
        return_tensors="pt"
    ).unsqueeze(0).to(model.device)

    return input_ids, image_tensor

def parse_yes_no(text):
    """
    Parse model output to extract yes/no prediction.
    Returns 1 for yes, 0 for no, -1 if unable to parse.
    """
    text = text.lower().strip()

    # Direct match at start
    if text.startswith("yes"):
        return 1
    if text.startswith("no"):
        return 0

    # Check first word
    first_word = text.split()[0] if text.split() else ""
    first_word = first_word.rstrip(".,!?")
    if first_word == "yes":
        return 1
    if first_word == "no":
        return 0

    # Search in text
    if "yes" in text and "no" not in text:
        return 1
    if "no" in text and "yes" not in text:
        return 0

    return -1  # Unable to parse
```

[15]:
```
@torch.no_grad()
def generate_answer(image_path, question, tokenizer, image_processor, model,␣
 ↪max_new_tokens=10, return_debug=False):
    """
    Generate an answer for a given image and question.
    """
    input_ids, image_tensor = prepare_inputs(
        image_path, question, tokenizer, image_processor, model
    )

    # Generate
    with torch.inference_mode():
        output_ids = model.generate(
```

```
            input_ids,
            images=image_tensor,
            do_sample=False,
            max_new_tokens=max_new_tokens,
            use_cache=True,
            pad_token_id=tokenizer.pad_token_id
        )

    prompt_len = int(input_ids.shape[1])
    output_len = int(output_ids.shape[1])
    output_includes_prompt = output_len >= prompt_len

    if output_includes_prompt:
        decoded_new_raw = tokenizer.decode(
            output_ids[0, prompt_len:],
            skip_special_tokens=False
        )
        generated = tokenizer.decode(
            output_ids[0, prompt_len:],
            skip_special_tokens=True
        ).strip()
        new_tokens = output_len - prompt_len
    else:
        # Some LLaVA generate paths return only new tokens (no prompt).
        decoded_new_raw = tokenizer.decode(
            output_ids[0],
            skip_special_tokens=False
        )
        generated = tokenizer.decode(
            output_ids[0],
            skip_special_tokens=True
        ).strip()
        new_tokens = output_len

    if not return_debug:
        return generated

    return {
        "generated": generated,
        "decoded_new_raw": decoded_new_raw,
        "prompt_len": prompt_len,
        "output_len": output_len,
        "new_tokens": new_tokens,
        "output_includes_prompt": output_includes_prompt,
    }
```

[16]:
```
# Test inference on a sample
test_sample = test_samples[0]
print(f"Testing inference...")
print(f"Question: {test_sample['question']}")
print(f"True answer: {test_sample['answer_text']}")

response = generate_answer(
    test_sample["image_path"],
    test_sample["question"],
    tokenizer, image_processor, model
)
print(f"Model response: {response}")
print(f"Parsed: {'yes' if parse_yes_no(response) == 1 else 'no' if␣
 ↪parse_yes_no(response) == 0 else 'unparseable'}")
```

```
Testing inference…
Question: Is there a pneumothorax?
True answer: yes
Model response: No, there is no pneumothorax
Parsed: no
```

1.6 6. Training Dataset Preparation

[17]:
```
from torch.utils.data import Dataset, DataLoader

class VQARadTrainDataset(Dataset):
    """Dataset for training with LLaVA-Med."""

    def __init__(self, samples, tokenizer, image_processor, model_config,␣
 ↪max_length=512):
        self.samples = samples
        self.tokenizer = tokenizer
        self.image_processor = image_processor
        self.model_config = model_config
        self.max_length = max_length

    def __len__(self):
        return len(self.samples)

    def __getitem__(self, idx):
        sample = self.samples[idx]

        # Load and process image
        image = Image.open(sample["image_path"]).convert("RGB")
        image_tensor = process_images([image], self.image_processor, self.
 ↪model_config)[0]
```

```
        # Create conversation with answer
        conv = conv_templates[CONV_MODE].copy()
        prompt = create_prompt(sample["question"])
        inp = DEFAULT_IMAGE_TOKEN + "\n" + prompt

        conv.append_message(conv.roles[0], inp)
        conv.append_message(conv.roles[1], sample["answer_text"])

        full_text = conv.get_prompt()

        # Tokenize full conversation
        input_ids = tokenizer_image_token(
            full_text,
            self.tokenizer,
            IMAGE_TOKEN_INDEX,
            return_tensors="pt"
        )

        # Create labels (mask everything except the answer)
        labels = input_ids.clone()

        # Find where the answer starts and mask everything before
        # The answer is after the assistant's turn marker
        conv_no_answer = conv_templates[CONV_MODE].copy()
        conv_no_answer.append_message(conv_no_answer.roles[0], inp)
        conv_no_answer.append_message(conv_no_answer.roles[1], None)
        prompt_only = conv_no_answer.get_prompt()

        prompt_ids = tokenizer_image_token(
            prompt_only,
            self.tokenizer,
            IMAGE_TOKEN_INDEX,
            return_tensors="pt"
        )
        prompt_len = prompt_ids.shape[0]

        # Mask prompt in labels
        labels[:prompt_len] = -100

        return {
            "input_ids": input_ids,
            "labels": labels,
            "images": image_tensor
        }
```

[18]:
```
def collate_fn(batch):
    """Collate function for DataLoader."""
```

```
    # Pad input_ids and labels
    max_len = max(item["input_ids"].shape[0] for item in batch)

    input_ids_list = []
    labels_list = []
    images_list = []
    attention_mask_list = []

    for item in batch:
        seq_len = item["input_ids"].shape[0]
        pad_len = max_len - seq_len

        # Pad input_ids with pad_token_id
        padded_input_ids = torch.cat([
            item["input_ids"],
            torch.full((pad_len,), tokenizer.pad_token_id, dtype=torch.long)
        ])
        input_ids_list.append(padded_input_ids)

        # Pad labels with -100
        padded_labels = torch.cat([
            item["labels"],
            torch.full((pad_len,), -100, dtype=torch.long)
        ])
        labels_list.append(padded_labels)

        # Attention mask
        attention_mask = torch.cat([
            torch.ones(seq_len, dtype=torch.long),
            torch.zeros(pad_len, dtype=torch.long)
        ])
        attention_mask_list.append(attention_mask)

        images_list.append(item["images"])

    return {
        "input_ids": torch.stack(input_ids_list),
        "labels": torch.stack(labels_list),
        "attention_mask": torch.stack(attention_mask_list),
        "images": torch.stack(images_list)
    }
```

[19]:
```
# Create datasets
train_dataset = VQARadTrainDataset(
    train_samples, tokenizer, image_processor, model.config
)
```

```
val_dataset = VQARadTrainDataset(
    val_samples, tokenizer, image_processor, model.config
)

print(f"Train dataset: {len(train_dataset)} samples")
print(f"Val dataset: {len(val_dataset)} samples")
```

```
Train dataset: 365 samples
Val dataset: 49 samples
```

1.7 7. Training Loop

[20]:
```
# Training hyperparameters
BATCH_SIZE = 1
GRADIENT_ACCUMULATION = 16  # Effective batch size = 16
LEARNING_RATE = 2e-5
NUM_EPOCHS = 10
WARMUP_RATIO = 0.1
WEIGHT_DECAY = 0.01
PATIENCE = 3  # For early stopping

# DataLoaders
train_loader = DataLoader(
    train_dataset,
    batch_size=BATCH_SIZE,
    shuffle=True,
    collate_fn=collate_fn,
    num_workers=0
)

val_loader = DataLoader(
    val_dataset,
    batch_size=BATCH_SIZE,
    shuffle=False,
    collate_fn=collate_fn,
    num_workers=0
)

print(f"Train batches: {len(train_loader)}")
print(f"Val batches: {len(val_loader)}")
```

```
Train batches: 365
Val batches: 49
```

[21]:
```
from torch.optim import AdamW
from torch.optim.lr_scheduler import CosineAnnealingLR

# Optimizer
```

```
optimizer = AdamW(
    filter(lambda p: p.requires_grad, model.parameters()),
    lr=LEARNING_RATE,
    weight_decay=WEIGHT_DECAY
)

# Scheduler
total_steps = len(train_loader) * NUM_EPOCHS // GRADIENT_ACCUMULATION
warmup_steps = int(total_steps * WARMUP_RATIO)
scheduler = CosineAnnealingLR(optimizer, T_max=total_steps)
```

[22]:
```
def train_epoch(model, loader, optimizer, scheduler, gradient_accumulation):
    """Train for one epoch."""
    model.train()
    total_loss = 0
    optimizer.zero_grad()

    pbar = tqdm(loader, desc="Training")
    for step, batch in enumerate(pbar):
        input_ids = batch["input_ids"].to(model.device)
        labels = batch["labels"].to(model.device)
        attention_mask = batch["attention_mask"].to(model.device)
        images = batch["images"].to(model.device, dtype=torch.float16)

        outputs = model(
            input_ids=input_ids,
            attention_mask=attention_mask,
            labels=labels,
            images=images
        )

        loss = outputs.loss / gradient_accumulation
        loss.backward()

        total_loss += loss.item() * gradient_accumulation

        if (step + 1) % gradient_accumulation == 0:
            torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
            optimizer.step()
            scheduler.step()
            optimizer.zero_grad()

        pbar.set_postfix({"loss": f"{total_loss / (step + 1):.4f}"})

    return total_loss / len(loader)

@torch.no_grad()
```

```
def evaluate_loss(model, loader):
    """Evaluate loss on validation set."""
    model.eval()
    total_loss = 0

    for batch in tqdm(loader, desc="Evaluating"):
        input_ids = batch["input_ids"].to(model.device)
        labels = batch["labels"].to(model.device)
        attention_mask = batch["attention_mask"].to(model.device)
        images = batch["images"].to(model.device, dtype=torch.float16)

        outputs = model(
            input_ids=input_ids,
            attention_mask=attention_mask,
            labels=labels,
            images=images
        )

        total_loss += outputs.loss.item()

    return total_loss / len(loader)
```

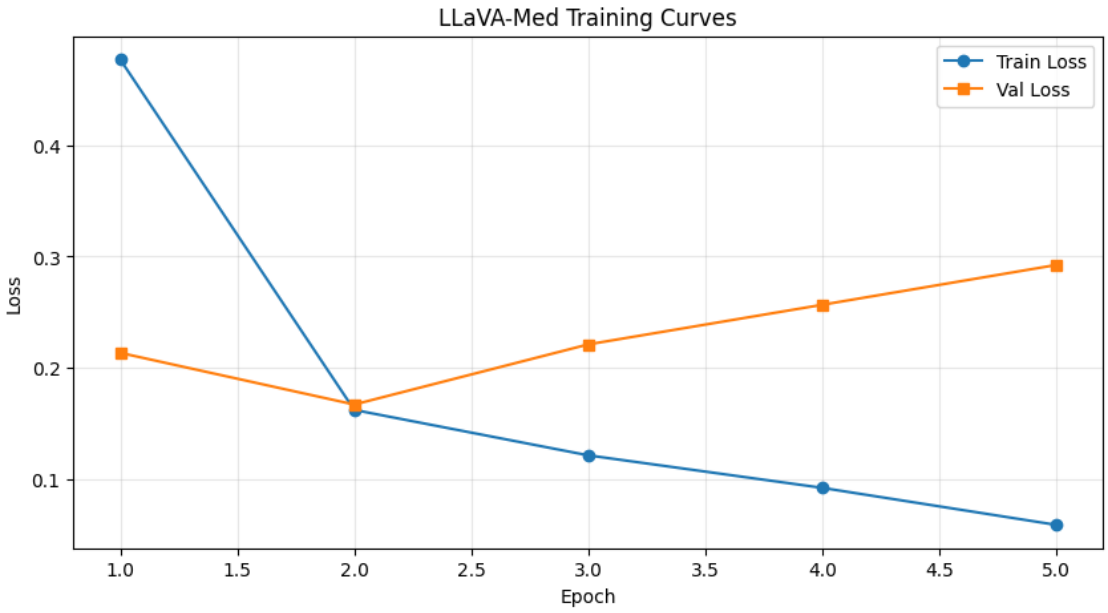
[23]:
```
# Training loop with early stopping
best_val_loss = float("inf")
patience_counter = 0
train_losses = []
val_losses = []

print("Starting training...")
print("="*50)

for epoch in range(NUM_EPOCHS):
    print(f"\nEpoch {epoch + 1}/{NUM_EPOCHS}")

    # Train
    train_loss = train_epoch(model, train_loader, optimizer, scheduler,␣
 ↪GRADIENT_ACCUMULATION)
    train_losses.append(train_loss)

    # Validate
    val_loss = evaluate_loss(model, val_loader)
    val_losses.append(val_loss)

    print(f"Train Loss: {train_loss:.4f} | Val Loss: {val_loss:.4f}")

    # Check for improvement
    if val_loss < best_val_loss:
```

```
        best_val_loss = val_loss
        patience_counter = 0
        # Save best model
        model.save_pretrained("./llava_med_best")
        print("  -> New best model saved!")
    else:
        patience_counter += 1
        print(f"  -> No improvement (patience: {patience_counter}/{PATIENCE})")

    # Early stopping
    if patience_counter >= PATIENCE:
        print(f"\nEarly stopping triggered at epoch {epoch + 1}")
        break

print("\nTraining complete!")
```

```
Starting training…
==================================================

Epoch 1/10

Training:   0%|          | 0/365 [00:00<?, ?it/s]`use_cache=True` is
incompatible with gradient checkpointing. Setting `use_cache=False`.
Training: 100%|      | 365/365 [06:36<00:00,  1.09s/it, loss=0.4771]
Evaluating: 100%|      | 49/49 [00:18<00:00,  2.70it/s]

Train Loss: 0.4771 | Val Loss: 0.2136
  -> New best model saved!

Epoch 2/10

Training: 100%|      | 365/365 [06:34<00:00,  1.08s/it, loss=0.1624]
Evaluating: 100%|      | 49/49 [00:18<00:00,  2.71it/s]

Train Loss: 0.1624 | Val Loss: 0.1672
  -> New best model saved!

Epoch 3/10

Training: 100%|      | 365/365 [06:34<00:00,  1.08s/it, loss=0.1215]
Evaluating: 100%|      | 49/49 [00:18<00:00,  2.71it/s]

Train Loss: 0.1215 | Val Loss: 0.2212
  -> No improvement (patience: 1/3)

Epoch 4/10

Training: 100%|      | 365/365 [06:33<00:00,  1.08s/it, loss=0.0923]
Evaluating: 100%|      | 49/49 [00:17<00:00,  2.75it/s]

Train Loss: 0.0923 | Val Loss: 0.2568
  -> No improvement (patience: 2/3)
```

```
Epoch 5/10

Training: 100%|      | 365/365 [06:28<00:00,  1.06s/it, loss=0.0591]
Evaluating: 100%|      | 49/49 [00:17<00:00,  2.76it/s]

Train Loss: 0.0591 | Val Loss: 0.2925
  -> No improvement (patience: 3/3)

Early stopping triggered at epoch 5

Training complete!
```

[24]:
```
# Plot training curves
plt.figure(figsize=(10, 5))
plt.plot(range(1, len(train_losses)+1), train_losses, label="Train Loss",␣
  ↪marker="o")
plt.plot(range(1, len(val_losses)+1), val_losses, label="Val Loss", marker="s")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("LLaVA-Med Training Curves")
plt.legend()
plt.grid(True, alpha=0.3)
plt.savefig("llava_med_training_curves.png", dpi=150)
plt.show()
```



1.8 8. Load Best Model and Evaluate

[25]:
```
# Load best model
from peft import PeftModel

# Reload base model (same 4-bit quantization settings)
tokenizer, base_model, image_processor, context_len = ␣
 ↪load_pretrained_model(**MODEL_LOAD_KWARGS)

# Load LoRA weights
model = PeftModel.from_pretrained(base_model, "./llava_med_best")
model.eval()
print("Best model loaded!")
```

```
Loading checkpoint shards:   0%|          | 0/4 [00:00<?, ?it/s]
```

```
Some weights of the model checkpoint at microsoft/llava-med-v1.5-mistral-7b were
not used when initializing LlavaMistralForCausalLM:
['model.vision_tower.vision_tower.vision_model.embeddings.class_embedding', 'mod
el.vision_tower.vision_tower.vision_model.embeddings.patch_embedding.weight', 'm
odel.vision_tower.vision_tower.vision_model.embeddings.position_embedding.weight
', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_norm1.bi
as', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_norm1.
weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_no
rm2.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_
norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.0.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.0.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.0.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.0.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.0.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.1.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.1.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.1.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.1.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.
```

k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.q_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.v_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.v_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.layer_norm1.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.layer_norm1.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.layer_norm2.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.k_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.q_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.v_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.v_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm1.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm1.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm2.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.k_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.q_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.v_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.v_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm1.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm1.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm2.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm2.weight',

'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
2.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.12.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.12.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.12.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.12.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.12.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.13.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
3.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.13.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.13.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.13.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.13.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.13.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.14.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
4.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.14.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.14.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.14.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.14.self_attn.v_proj.bias', 'model.vision_

tower.vision_tower.vision_model.encoder.layers.14.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.15.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
5.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.15.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.15.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.15.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.15.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.15.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.16.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
6.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.16.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.16.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.16.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.16.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.16.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.17.self_att

n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
7.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.17.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.17.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.17.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.17.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.17.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.18.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
8.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.18.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.18.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.18.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.18.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.18.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.19.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
9.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.19.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.19.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.19.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.19.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.19.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm1.bias', 'm
odel.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm1.weight'
, 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm2.bia
s', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm2.w
eight',

'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.2.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.2.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.2.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.2.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.2.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.20.layer_norm1.bias', 'model.vi
sion_tower.vision_tower.vision_model.encoder.layers.20.layer_norm1.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.20.layer_norm2.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.20.layer_norm2.weigh
t',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.20.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
0.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.20.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.20.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.20.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.20.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.20.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.21.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
1.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.21.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.21.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.21.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.21.self_attn.v_proj.bias', 'model.vision_

tower.vision_tower.vision_model.encoder.layers.21.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.22.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
2.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.22.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.22.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.22.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.22.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.22.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.23.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
3.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.23.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.23.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.23.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.23.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.23.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm1.bias', 'm
odel.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm1.weight'
, 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm2.bia
s', 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm2.w
eight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.self_attn.

k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.3.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.3.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.3.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.3.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.3.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.4.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.4.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.4.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.4.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.4.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.4.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.4.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.4.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.4.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.4.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.4.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.5.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.5.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.5.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.5.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.5.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.5.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.5.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.5.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.5.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.5.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.5.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.6.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.6.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.6.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.6.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc2.bias',

'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.6.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.6.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.6.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.6.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.6.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.6.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.6.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.7.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.7.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.7.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.7.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.7.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.7.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.7.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.7.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.7.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.7.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.7.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.8.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.8.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.8.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.8.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.8.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.8.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.8.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.8.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.8.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.8.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.8.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.9.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.9.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.9.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.9.layer_norm2.weight',

```
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.9.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.9.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.9.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.9.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.9.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.9.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.9.self_attn.v_proj.weight',
'model.vision_tower.vision_tower.vision_model.post_layernorm.bias',
'model.vision_tower.vision_tower.vision_model.post_layernorm.weight',
'model.vision_tower.vision_tower.vision_model.pre_layrnorm.bias',
'model.vision_tower.vision_tower.vision_model.pre_layrnorm.weight']
- This IS expected if you are initializing LlavaMistralForCausalLM from the
checkpoint of a model trained on another task or with another architecture (e.g.
initializing a BertForSequenceClassification model from a BertForPreTraining
model).
- This IS NOT expected if you are initializing LlavaMistralForCausalLM from the
checkpoint of a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).
Some parameters are on the meta device because they were offloaded to the cpu.

Best model loaded!
```

[26]:
```python
@torch.no_grad()
def evaluate_accuracy(samples, desc="Evaluating", debug=False, debug_n=5):
    """
    Evaluate the model on a set of samples.
    Returns predictions, labels, and raw outputs.
    """
    model.eval()

    predictions = []
    labels = []
    raw_outputs = []
    unparseable = 0

    import inspect
    supports_return_debug = "return_debug" in inspect.
 ↪signature(generate_answer).parameters
    if debug and not supports_return_debug:
        print("[DEBUG] generate_answer does not support return_debug; re-run␣
 ↪its definition cell to enable raw token logging.")
```

```
    for i, sample in enumerate(tqdm(samples, desc=desc)):
        # Generate answer
        if debug and i < debug_n and supports_return_debug:
            response_debug = generate_answer(
                sample["image_path"],
                sample["question"],
                tokenizer, image_processor, model,
                return_debug=True
            )
            response = response_debug["generated"]
        else:
            response_debug = None
            response = generate_answer(
                sample["image_path"],
                sample["question"],
                tokenizer, image_processor, model
            )

        raw_outputs.append(response)

        # Parse prediction
        pred = parse_yes_no(response)
        if pred == -1:
            unparseable += 1
            pred = 0  # Default to 'no' for unparseable

        predictions.append(pred)
        labels.append(sample["label"])

        if debug and i < debug_n:
            true_label = sample.get("answer_text")
            if true_label is None:
                true_label = "yes" if sample["label"] == 1 else "no"
            if response_debug is not None:
                raw = response_debug["decoded_new_raw"]
                raw_preview = raw[:200] + ("..." if len(raw) > 200 else "")
                prompt_len = response_debug["prompt_len"]
                output_len = response_debug["output_len"]
                new_tokens = response_debug["new_tokens"]
                output_includes_prompt =␣
↪response_debug["output_includes_prompt"]
            else:
                raw_preview = "(unavailable: return_debug not supported)"
                prompt_len = "?"
                output_len = "?"
                new_tokens = "?"
                output_includes_prompt = "?"
```

```
            print(
                f"[DEBUG {desc} #{i}] question={sample.get('question', '')!r} "
                f"true={true_label} pred={pred} generated={response!r} "
                f"prompt_len={prompt_len} "
                f"output_len={output_len} "
                f"new_tokens={new_tokens} "
                f"output_includes_prompt={output_includes_prompt} "
                f"raw_preview={raw_preview!r}"
            )

    if unparseable > 0:
        print(f"Warning: {unparseable} outputs could not be parsed␣
↪({unparseable/len(samples)*100:.1f}%)")

    return predictions, labels, raw_outputs
```

[27]:
```
# Evaluate on validation set
print("Evaluating on validation set...")
DEBUG_EVAL = True
DEBUG_EVAL_N = 5
val_preds, val_labels, val_outputs = evaluate_accuracy(val_samples, "Val",␣
↪debug=DEBUG_EVAL, debug_n=DEBUG_EVAL_N)

val_acc = accuracy_score(val_labels, val_preds)
val_f1 = f1_score(val_labels, val_preds, average="macro")

print(f"\nValidation Results:")
print(f"  Accuracy: {val_acc:.4f}")
print(f"  Macro-F1: {val_f1:.4f}")
```

```
Evaluating on validation set…

Val:   2%|          | 1/49 [00:00<00:35,  1.35it/s]

[DEBUG Val #0] question='Is this a lateral film?' true=no pred=0 generated='no'
prompt_len=70 output_len=3 new_tokens=3 output_includes_prompt=False
raw_preview='no </s>'

Val:   4%|          | 2/49 [00:01<00:29,  1.58it/s]

[DEBUG Val #1] question='Are the lungs normal?' true=no pred=0 generated='no'
prompt_len=68 output_len=3 new_tokens=3 output_includes_prompt=False
raw_preview='no </s>'

Val:   6%|          | 3/49 [00:01<00:29,  1.56it/s]

[DEBUG Val #2] question='Is the chest x-ray normal?' true=no pred=0
generated='no' prompt_len=71 output_len=3 new_tokens=3
output_includes_prompt=False raw_preview='no </s>'

Val:   8%|          | 4/49 [00:02<00:27,  1.64it/s]
```

```
[DEBUG Val #3] question='Is the right costophrenic angle sharp?' true=no pred=1
generated='yes' prompt_len=73 output_len=3 new_tokens=3
output_includes_prompt=False raw_preview='yes </s>'

Val:  10%|          | 5/49 [00:03<00:26,  1.67it/s]

[DEBUG Val #4] question='Is the right costophrenic angle easily visualized?'
true=no pred=1 generated='yes' prompt_len=75 output_len=3 new_tokens=3
output_includes_prompt=False raw_preview='yes </s>'

Val: 100%|      | 49/49 [00:27<00:00,  1.76it/s]


Validation Results:
  Accuracy: 0.7347
  Macro-F1: 0.6817
```

[28]:
```python
# Evaluate on test set
print("\nEvaluating on test set...")
DEBUG_EVAL = True
DEBUG_EVAL_N = 5
test_preds, test_labels, test_outputs = evaluate_accuracy(test_samples, "Test",␣
 ↪debug=DEBUG_EVAL, debug_n=DEBUG_EVAL_N)

test_acc = accuracy_score(test_labels, test_preds)
test_f1 = f1_score(test_labels, test_preds, average="macro")

print(f"\nTest Results:")
print(f"  Accuracy: {test_acc:.4f}")
print(f"  Macro-F1: {test_f1:.4f}")
```

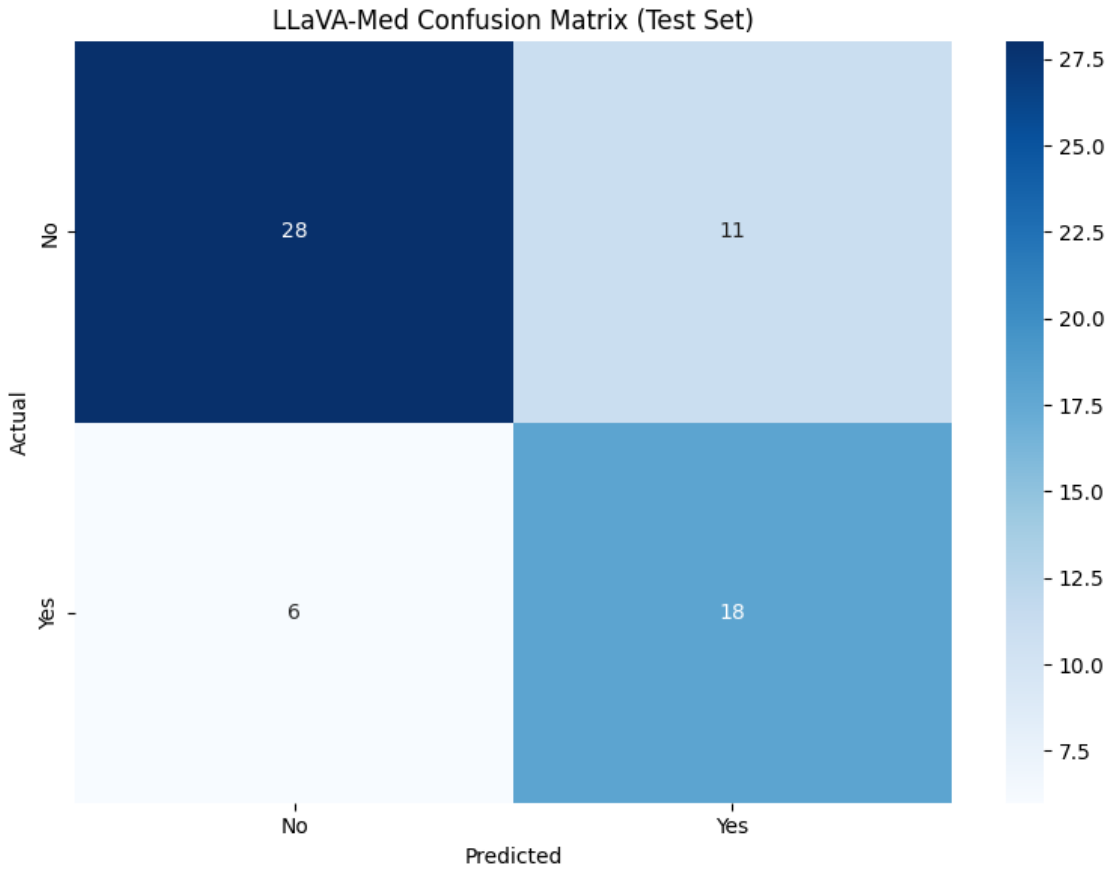
```
Evaluating on test set…

Test:   2%|          | 1/63 [00:00<00:35,  1.73it/s]

[DEBUG Test #0] question='Is there a pneumothorax?' true=yes pred=0
generated='no' prompt_len=72 output_len=3 new_tokens=3
output_includes_prompt=False raw_preview='no </s>'

Test:   3%|          | 2/63 [00:01<00:34,  1.75it/s]

[DEBUG Test #1] question='Do you see a pleural effusion?' true=no pred=0
generated='no' prompt_len=72 output_len=3 new_tokens=3
output_includes_prompt=False raw_preview='no </s>'

Test:   5%|          | 3/63 [00:01<00:34,  1.75it/s]

[DEBUG Test #2] question='Is there a pleural effusion present?' true=no pred=0
generated='no' prompt_len=72 output_len=3 new_tokens=3
output_includes_prompt=False raw_preview='no </s>'
```

```
Test:   6%|          | 4/63 [00:02<00:33,  1.77it/s]

[DEBUG Test #3] question='Do you see cardiomegaly?' true=no pred=1
generated='yes' prompt_len=72 output_len=3 new_tokens=3
output_includes_prompt=False raw_preview='yes </s>'

Test:   8%|          | 5/63 [00:02<00:32,  1.76it/s]

[DEBUG Test #4] question='Is cardiomegaly present?' true=no pred=1
generated='yes' prompt_len=71 output_len=3 new_tokens=3
output_includes_prompt=False raw_preview='yes </s>'

Test: 100%|      | 63/63 [00:36<00:00,  1.72it/s]


Test Results:
  Accuracy: 0.7302
  Macro-F1: 0.7232
```

[29]:
```
# Detailed classification report
print("\nClassification Report (Test Set):")
print(classification_report(test_labels, test_preds, target_names=["No",␣
 ↪"Yes"]))
```

```
Classification Report (Test Set):
              precision    recall  f1-score   support

          No       0.82      0.72      0.77        39
         Yes       0.62      0.75      0.68        24

    accuracy                           0.73        63
   macro avg       0.72      0.73      0.72        63
weighted avg       0.75      0.73      0.73        63
```

[30]:
```
# Confusion matrix
cm = confusion_matrix(test_labels, test_preds)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["No", "Yes"], yticklabels=["No", "Yes"])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("LLaVA-Med Confusion Matrix (Test Set)")
plt.tight_layout()
plt.savefig("llava_med_confusion_matrix.png", dpi=150)
plt.show()
```



1.9 9. Error Analysis

[31]:
```
# Analyze errors
errors = []
for i, (pred, label, output, sample) in enumerate(zip(test_preds, test_labels,␣
 ↪test_outputs, test_samples)):
    if pred != label:
        errors.append({
            "question": sample["question"],
            "true_label": "yes" if label == 1 else "no",
            "predicted": "yes" if pred == 1 else "no",
            "raw_output": output,
            "image_id": sample["image_id"]
        })

print(f"Total errors: {len(errors)} / {len(test_samples)} ({len(errors)/
 ↪len(test_samples)*100:.1f}%)")
print("\nSample errors:")
for e in errors[:10]:
```

```
    print(f"  Q: {e['question']}")
    print(f"  True: {e['true_label']}, Pred: {e['predicted']}, Raw:␣
 ↪'{e['raw_output']}'")
    print()
```

```
Total errors: 17 / 63 (27.0%)

Sample errors:
  Q: Is there a pneumothorax?
  True: yes, Pred: no, Raw: 'no'

  Q: Do you see cardiomegaly?
  True: no, Pred: yes, Raw: 'yes'

  Q: Is cardiomegaly present?
  True: no, Pred: yes, Raw: 'yes'

  Q: Is there enlargement of the pulmonary vasculature?
  True: no, Pred: yes, Raw: 'yes'

  Q: Is there a lung mass?
  True: yes, Pred: no, Raw: 'no'

  Q: Is there a mass in the lung?
  True: yes, Pred: no, Raw: 'no'

  Q: Is the heart enlarged?
  True: no, Pred: yes, Raw: 'yes'

  Q: Is the heart size increased?
  True: no, Pred: yes, Raw: 'yes'

  Q: Is the cardiac silhouette enlarged?
  True: no, Pred: yes, Raw: 'yes'

  Q: Is the mediastinum widened?
  True: yes, Pred: no, Raw: 'no'
```

[32]:
```
# Analyze error patterns
negation_keywords = ["no ", "not ", "without", "absence", "negative", "isn't",␣
 ↪"aren't"]
spatial_keywords = ["left", "right", "upper", "lower", "middle", "bilateral",␣
 ↪"base", "apex"]

if len(errors) > 0:
```

```
    negation_errors = sum(1 for e in errors if any(kw in e["question"].lower()␣
 ↪for kw in negation_keywords))
    spatial_errors = sum(1 for e in errors if any(kw in e["question"].lower()␣
 ↪for kw in spatial_keywords))

    print(f"\nError Analysis:")
    print(f"  Errors involving negation: {negation_errors} ({negation_errors/
 ↪len(errors)*100:.1f}% of errors)")
    print(f"  Errors involving spatial terms: {spatial_errors} ({spatial_errors/
 ↪len(errors)*100:.1f}% of errors)")
else:
    print("No errors to analyze!")
```

```
Error Analysis:
  Errors involving negation: 0 (0.0% of errors)
  Errors involving spatial terms: 1 (5.9% of errors)
```

1.10 10. Comparison with CNN Baseline

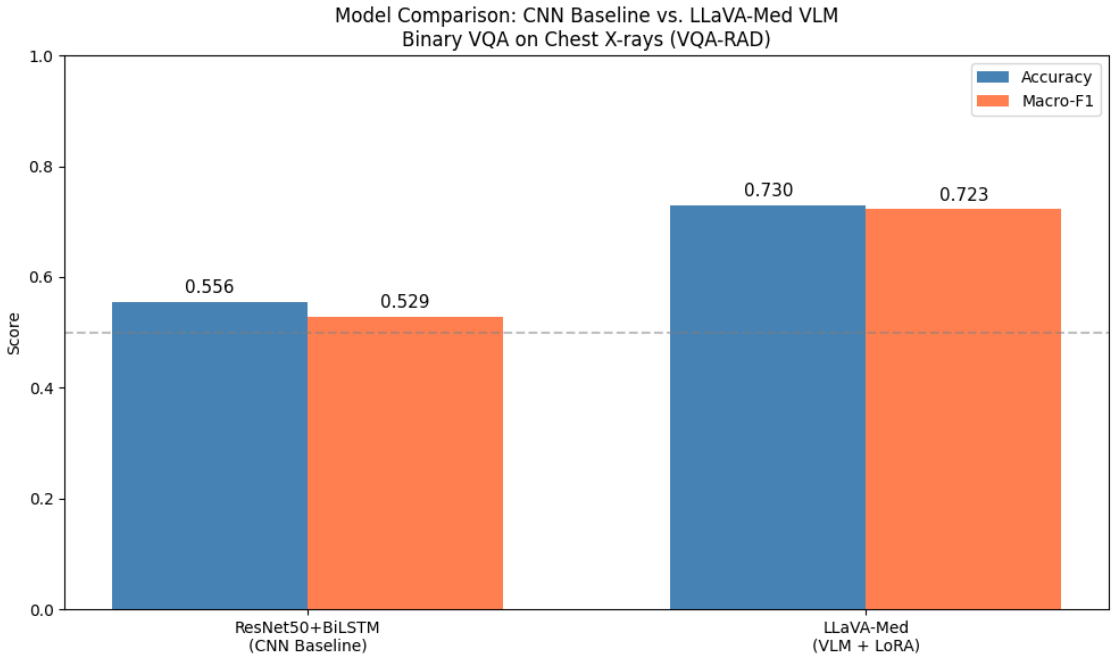
[33]:
```
# CNN Baseline results (from teammate's notebook AA.ipynb)
cnn_results = {
    "model": "ResNet50 + BiLSTM",
    "test_acc": 0.5556,
    "test_f1": 0.5288
}

# LLaVA-Med results
vlm_results = {
    "model": "LLaVA-Med (LoRA)",
    "test_acc": test_acc,
    "test_f1": test_f1
}

print("="*65)
print("COMPARISON: CNN Baseline vs. LLaVA-Med VLM")
print("="*65)
print(f"\n{'Model':<25} {'Test Accuracy':<15} {'Test Macro-F1':<15}")
print("-"*55)
print(f"{cnn_results['model']:<25} {cnn_results['test_acc']:<15.4f}␣
 ↪{cnn_results['test_f1']:<15.4f}")
print(f"{vlm_results['model']:<25} {vlm_results['test_acc']:<15.4f}␣
 ↪{vlm_results['test_f1']:<15.4f}")
print("-"*55)

acc_diff = vlm_results['test_acc'] - cnn_results['test_acc']
f1_diff = vlm_results['test_f1'] - cnn_results['test_f1']
```

```
print(f"{'Difference':<25} {acc_diff:+.4f}          {f1_diff:+.4f}")
print("="*65)
```

```
================================================================
COMPARISON: CNN Baseline vs. LLaVA-Med VLM
================================================================

Model                     Test Accuracy   Test Macro-F1
----------------------------------------------------------
ResNet50 + BiLSTM         0.5556          0.5288
LLaVA-Med (LoRA)          0.7302          0.7232
----------------------------------------------------------
Difference                +0.1746          +0.1944
================================================================
```

[34]:
```
# Visualization of comparison
models = ["ResNet50+BiLSTM\n(CNN Baseline)", "LLaVA-Med\n(VLM + LoRA)"]
accuracy = [cnn_results["test_acc"], vlm_results["test_acc"]]
f1_scores = [cnn_results["test_f1"], vlm_results["test_f1"]]

x = np.arange(len(models))
width = 0.35

fig, ax = plt.subplots(figsize=(10, 6))
bars1 = ax.bar(x - width/2, accuracy, width, label="Accuracy",␣
 ↪color="steelblue")
bars2 = ax.bar(x + width/2, f1_scores, width, label="Macro-F1", color="coral")

ax.set_ylabel("Score")
ax.set_title("Model Comparison: CNN Baseline vs. LLaVA-Med VLM\nBinary VQA on␣
 ↪Chest X-rays (VQA-RAD)")
ax.set_xticks(x)
ax.set_xticklabels(models)
ax.legend()
ax.set_ylim(0, 1)

# Add value labels
for bar in bars1:
    height = bar.get_height()
    ax.annotate(f'{height:.3f}',
                xy=(bar.get_x() + bar.get_width()/2, height),
                xytext=(0, 3), textcoords="offset points",
                ha='center', va='bottom', fontsize=11)

for bar in bars2:
    height = bar.get_height()
    ax.annotate(f'{height:.3f}',
```

```
                xy=(bar.get_x() + bar.get_width()/2, height),
                xytext=(0, 3), textcoords="offset points",
                ha='center', va='bottom', fontsize=11)

plt.axhline(y=0.5, color='gray', linestyle='--', alpha=0.5, label='Random␣
 ↪baseline')
plt.tight_layout()
plt.savefig("model_comparison.png", dpi=150)
plt.show()
```



1.11 11. Zero-Shot Evaluation (Baseline)

Evaluate LLaVA-Med WITHOUT fine-tuning to measure the benefit of LoRA adaptation.

[35]:
```
# Load base model (without LoRA) for zero-shot evaluation
print("Loading base LLaVA-Med for zero-shot evaluation...")

tokenizer_zs, model_zs, image_processor_zs, _ =␣
 ↪load_pretrained_model(**MODEL_LOAD_KWARGS)
model_zs.eval()

print("Base model loaded!")
```

```
Loading base LLaVA-Med for zero-shot evaluation…

Loading checkpoint shards:   0%|          | 0/4 [00:00<?, ?it/s]
```

```
Some weights of the model checkpoint at microsoft/llava-med-v1.5-mistral-7b were
not used when initializing LlavaMistralForCausalLM:
['model.vision_tower.vision_tower.vision_model.embeddings.class_embedding', 'mod
el.vision_tower.vision_tower.vision_model.embeddings.patch_embedding.weight', 'm
odel.vision_tower.vision_tower.vision_model.embeddings.position_embedding.weight
', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_norm1.bi
as', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_norm1.
weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_no
rm2.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.layer_
norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.0.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.0.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.0.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.0.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.0.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.0.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.0.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.1.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.1.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.1.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.1.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.1.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.1.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.1.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.1.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.1.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.10.layer_norm1.bias', 'model.vi
sion_tower.vision_tower.vision_model.encoder.layers.10.layer_norm1.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.10.layer_norm2.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.10.layer_norm2.weigh
t',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc2.bias',
```

'model.vision_tower.vision_tower.vision_model.encoder.layers.10.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.10.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
0.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.10.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.10.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.10.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.10.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.10.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.11.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
1.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.11.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.11.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.11.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.11.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.11.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.12.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.12.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
2.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.12.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.12.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.12.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.12.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.12.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_norm1.weigh

t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.13.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.13.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.13.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
3.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.13.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.13.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.13.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.13.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.13.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.14.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.14.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.14.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
4.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.14.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.14.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.14.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.14.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.14.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.15.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.15.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.15.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
5.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.15.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_

model.encoder.layers.15.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.15.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.15.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.15.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.16.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.16.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.16.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
6.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.16.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.16.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.16.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.16.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.16.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.17.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.17.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.17.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
7.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.17.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.17.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.17.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.17.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.17.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.18.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc2.bias',

'model.vision_tower.vision_tower.vision_model.encoder.layers.18.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.18.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.18.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
8.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.18.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.18.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.18.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.18.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.18.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.19.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.19.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.19.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.1
9.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.19.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.19.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.19.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.19.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.19.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm1.bias', 'm
odel.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm1.weight'
, 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm2.bia
s', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.layer_norm2.w
eight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.2.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.2.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.2.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.2.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.2.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.2.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.20.layer_norm1.bias', 'model.vi
sion_tower.vision_tower.vision_model.encoder.layers.20.layer_norm1.weight', 'mod

el.vision_tower.vision_tower.vision_model.encoder.layers.20.layer_norm2.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.20.layer_norm2.weigh
t',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.20.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.20.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
0.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.20.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.20.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.20.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.20.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.20.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.21.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.21.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.21.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
1.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.21.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.21.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.21.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.21.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.21.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.22.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.22.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.22.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
2.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.22.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_

model.encoder.layers.22.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.22.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.22.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.22.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_norm1.bias', '
model.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_norm1.weigh
t', 'model.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_norm2.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.23.layer_nor
m2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.23.self_attn.k_proj
.bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.23.self_att
n.k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.2
3.self_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encode
r.layers.23.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_
model.encoder.layers.23.self_attn.q_proj.bias', 'model.vision_tower.vision_tower
.vision_model.encoder.layers.23.self_attn.q_proj.weight', 'model.vision_tower.vi
sion_tower.vision_model.encoder.layers.23.self_attn.v_proj.bias', 'model.vision_
tower.vision_tower.vision_model.encoder.layers.23.self_attn.v_proj.weight', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm1.bias', 'm
odel.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm1.weight'
, 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm2.bia
s', 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.layer_norm2.w
eight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.3.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.3.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.3.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.3.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.3.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.3.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.3.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.4.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.4.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.4.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.4.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.4.mlp.fc2.weight',

'model.vision_tower.vision_tower.vision_model.encoder.layers.4.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.4.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.4.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.4.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.4.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.4.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.4.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.4.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.5.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.5.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.5.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.5.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.5.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.5.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.5.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.5.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.5.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.5.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.5.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.5.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.6.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.6.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.6.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.6.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.6.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.6.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.6.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.6.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.6.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.6.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.6.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.6.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.7.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.7.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.7.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.7.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc1.bias',

'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.7.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.7.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.7.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.7.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.7.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.7.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.7.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.7.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.8.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.8.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.8.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.8.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.8.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.8.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.8.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.8.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.8.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.8.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.8.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.8.self_attn.v_proj.weight', 'model.visio
n_tower.vision_tower.vision_model.encoder.layers.9.layer_norm1.bias', 'model.vis
ion_tower.vision_tower.vision_model.encoder.layers.9.layer_norm1.weight', 'model
.vision_tower.vision_tower.vision_model.encoder.layers.9.layer_norm2.bias', 'mod
el.vision_tower.vision_tower.vision_model.encoder.layers.9.layer_norm2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc1.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc1.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc2.bias',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.mlp.fc2.weight',
'model.vision_tower.vision_tower.vision_model.encoder.layers.9.self_attn.k_proj.
bias', 'model.vision_tower.vision_tower.vision_model.encoder.layers.9.self_attn.
k_proj.weight', 'model.vision_tower.vision_tower.vision_model.encoder.layers.9.s
elf_attn.out_proj.bias', 'model.vision_tower.vision_tower.vision_model.encoder.l
ayers.9.self_attn.out_proj.weight', 'model.vision_tower.vision_tower.vision_mode
l.encoder.layers.9.self_attn.q_proj.bias', 'model.vision_tower.vision_tower.visi
on_model.encoder.layers.9.self_attn.q_proj.weight', 'model.vision_tower.vision_t
ower.vision_model.encoder.layers.9.self_attn.v_proj.bias', 'model.vision_tower.v
ision_tower.vision_model.encoder.layers.9.self_attn.v_proj.weight',
'model.vision_tower.vision_tower.vision_model.post_layernorm.bias',
'model.vision_tower.vision_tower.vision_model.post_layernorm.weight',

```
'model.vision_tower.vision_tower.vision_model.pre_layrnorm.bias',
'model.vision_tower.vision_tower.vision_model.pre_layrnorm.weight']
- This IS expected if you are initializing LlavaMistralForCausalLM from the
checkpoint of a model trained on another task or with another architecture (e.g.
initializing a BertForSequenceClassification model from a BertForPreTraining
model).
- This IS NOT expected if you are initializing LlavaMistralForCausalLM from the
checkpoint of a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).

Base model loaded!
```

[36]:
```python
@torch.no_grad()
def evaluate_zero_shot(samples, model_zs, tokenizer_zs, image_processor_zs,␣
↪desc="Zero-shot", debug=False, debug_n=5):
    """Evaluate zero-shot performance."""
    predictions = []
    labels = []
    raw_outputs = []
    unparseable = 0

    import inspect
    supports_return_debug = "return_debug" in inspect.
↪signature(generate_answer).parameters
    if debug and not supports_return_debug:
        print("[DEBUG] generate_answer does not support return_debug; re-run␣
↪its definition cell to enable raw token logging.")

    for i, sample in enumerate(tqdm(samples, desc=desc)):
        if debug and i < debug_n and supports_return_debug:
            response_debug = generate_answer(
                sample["image_path"],
                sample["question"],
                tokenizer_zs, image_processor_zs, model_zs,
                return_debug=True
            )
            response = response_debug["generated"]
        else:
            response_debug = None
            response = generate_answer(
                sample["image_path"],
                sample["question"],
                tokenizer_zs, image_processor_zs, model_zs
            )

        raw_outputs.append(response)

        pred = parse_yes_no(response)
```

```
        if pred == -1:
            unparseable += 1
            pred = 0

        predictions.append(pred)
        labels.append(sample["label"])

        if debug and i < debug_n:
            true_label = sample.get("answer_text")
            if true_label is None:
                true_label = "yes" if sample["label"] == 1 else "no"
            if response_debug is not None:
                raw = response_debug["decoded_new_raw"]
                raw_preview = raw[:200] + ("..." if len(raw) > 200 else "")
                prompt_len = response_debug["prompt_len"]
                output_len = response_debug["output_len"]
                new_tokens = response_debug["new_tokens"]
                output_includes_prompt = ␣
↪response_debug["output_includes_prompt"]
            else:
                raw_preview = "(unavailable: return_debug not supported)"
                prompt_len = "?"
                output_len = "?"
                new_tokens = "?"
                output_includes_prompt = "?"
            print(
                f"[DEBUG {desc} #{i}] question={sample.get('question', '')!r} "
                f"true={true_label} pred={pred} generated={response!r} "
                f"prompt_len={prompt_len} "
                f"output_len={output_len} "
                f"new_tokens={new_tokens} "
                f"output_includes_prompt={output_includes_prompt} "
                f"raw_preview={raw_preview!r}"
            )

    if unparseable > 0:
        print(f"Warning: {unparseable} outputs unparseable ({unparseable/
↪len(samples)*100:.1f}%)")

    return predictions, labels, raw_outputs
```

[37]:
```
# Zero-shot evaluation on test set
print("Zero-shot evaluation on test set...")
DEBUG_EVAL = True
DEBUG_EVAL_N = 5
zs_preds, zs_labels, zs_outputs = evaluate_zero_shot(
```

```
    test_samples, model_zs, tokenizer_zs, image_processor_zs, "Zero-shot Test",␣
  ↪debug=DEBUG_EVAL, debug_n=DEBUG_EVAL_N
)

zs_acc = accuracy_score(zs_labels, zs_preds)
zs_f1 = f1_score(zs_labels, zs_preds, average="macro")

print(f"\nZero-shot Test Results:")
print(f"  Accuracy: {zs_acc:.4f}")
print(f"  Macro-F1: {zs_f1:.4f}")
```

```
Zero-shot evaluation on test set…

Zero-shot Test:   2%|          | 1/63 [00:00<00:43,  1.42it/s]

[DEBUG Zero-shot Test #0] question='Is there a pneumothorax?' true=yes pred=0
generated='No, there is no pneumothorax' prompt_len=72 output_len=10
new_tokens=10 output_includes_prompt=False raw_preview='No, there is no
pneumothorax'

Zero-shot Test:   3%|          | 2/63 [00:01<00:42,  1.43it/s]

[DEBUG Zero-shot Test #1] question='Do you see a pleural effusion?' true=no
pred=1 generated='Yes, the chest x-ray shows a ple' prompt_len=72 output_len=10
new_tokens=10 output_includes_prompt=False raw_preview='Yes, the chest x-ray
shows a ple'

Zero-shot Test:   5%|          | 3/63 [00:02<00:42,  1.42it/s]

[DEBUG Zero-shot Test #2] question='Is there a pleural effusion present?'
true=no pred=1 generated='Yes, there is a pleural effusion present'
prompt_len=72 output_len=10 new_tokens=10 output_includes_prompt=False
raw_preview='Yes, there is a pleural effusion present'

Zero-shot Test:   6%|          | 4/63 [00:02<00:41,  1.42it/s]

[DEBUG Zero-shot Test #3] question='Do you see cardiomegaly?' true=no pred=1
generated='Yes, the Chest X-ray shows card' prompt_len=72 output_len=10
new_tokens=10 output_includes_prompt=False raw_preview='Yes, the Chest X-ray
shows card'

Zero-shot Test:   8%|          | 5/63 [00:03<00:42,  1.38it/s]

[DEBUG Zero-shot Test #4] question='Is cardiomegaly present?' true=no pred=1
generated='Yes, the Chest X-ray shows card' prompt_len=71 output_len=10
new_tokens=10 output_includes_prompt=False raw_preview='Yes, the Chest X-ray
shows card'

Zero-shot Test: 100%|      | 63/63 [00:51<00:00,  1.22it/s]


Zero-shot Test Results:
  Accuracy: 0.5238
  Macro-F1: 0.5026
```

[38]:
```
# Cleanup
del model_zs
torch.cuda.empty_cache()
```

1.12 12. Complete Results Summary

[39]:
```
print("="*70)
print("COMPLETE RESULTS SUMMARY")
print("Binary Medical VQA on Chest X-rays (VQA-RAD Dataset)")
print("="*70)

print(f"\nDataset Statistics:")
print(f"  Total filtered samples: {len(samples)}")
print(f"  Train: {len(train_samples)} | Val: {len(val_samples)} | Test:␣
 ↪{len(test_samples)}")

print(f"\n{'Model':<30} {'Test Acc':<12} {'Test F1':<12}")
print("-"*55)
print(f"{'ResNet50 + BiLSTM (Baseline)':<30} {cnn_results['test_acc']:.4f}      ␣
 ↪ {cnn_results['test_f1']:.4f}")
print(f"{'LLaVA-Med Zero-Shot':<30} {zs_acc:.4f}       {zs_f1:.4f}")
print(f"{'LLaVA-Med + LoRA Fine-tuned':<30} {test_acc:.4f}       {test_f1:.4f}")
print("-"*55)

print(f"\nKey Findings:")
if test_acc > cnn_results['test_acc']:
    print(f"  - LLaVA-Med (fine-tuned) outperforms CNN baseline by {(test_acc -␣
 ↪cnn_results['test_acc'])*100:.1f}% accuracy")
else:
    print(f"  - CNN baseline is competitive, with {(cnn_results['test_acc'] -␣
 ↪test_acc)*100:.1f}% higher accuracy")

print(f"  - Fine-tuning improves LLaVA-Med by {(test_acc - zs_acc)*100:.1f}%␣
 ↪over zero-shot")
print(f"  - LoRA enables efficient adaptation with minimal trainable␣
 ↪parameters")

print("="*70)
```

```
======================================================================
COMPLETE RESULTS SUMMARY
Binary Medical VQA on Chest X-rays (VQA-RAD Dataset)
======================================================================

Dataset Statistics:
  Total filtered samples: 477
```

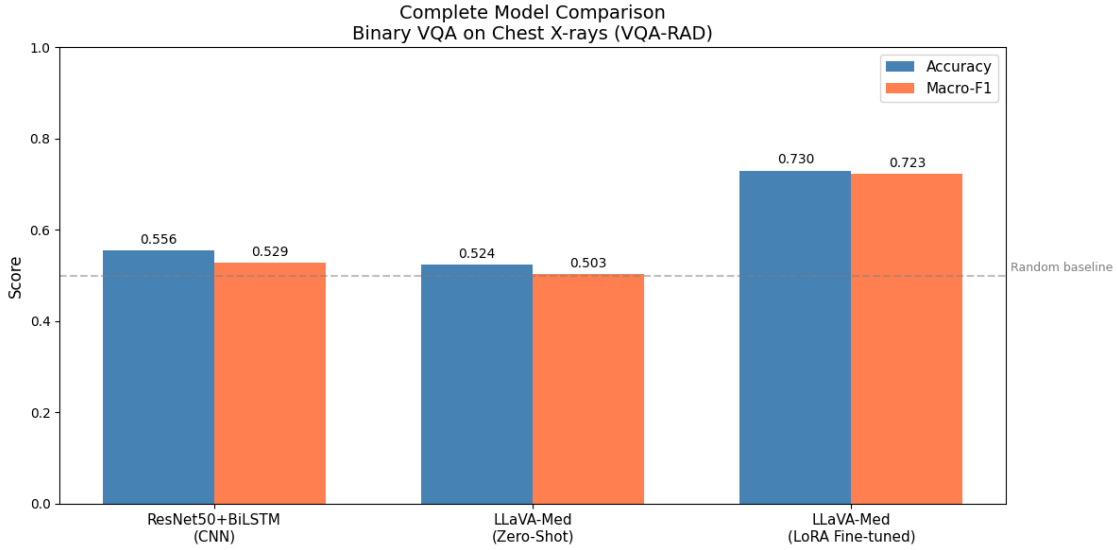
```
  Train: 365 | Val: 49 | Test: 63

Model                           Test Acc     Test F1
------------------------------------------------------------
ResNet50 + BiLSTM (Baseline)    0.5556       0.5288
LLaVA-Med Zero-Shot             0.5238       0.5026
LLaVA-Med + LoRA Fine-tuned     0.7302       0.7232
------------------------------------------------------------

Key Findings:
  - LLaVA-Med (fine-tuned) outperforms CNN baseline by 17.5% accuracy
  - Fine-tuning improves LLaVA-Med by 20.6% over zero-shot
  - LoRA enables efficient adaptation with minimal trainable parameters
======================================================================
```

[40]:
```python
# Save all results to JSON
results = {
    "dataset": "VQA-RAD (Chest X-ray, Binary)",
    "seed": SEED,
    "splits": {
        "train": len(train_samples),
        "val": len(val_samples),
        "test": len(test_samples)
    },
    "models": {
        "cnn_baseline": cnn_results,
        "llava_med_zero_shot": {
            "model": "LLaVA-Med v1.5 (Zero-Shot)",
            "test_acc": zs_acc,
            "test_f1": zs_f1
        },
        "llava_med_finetuned": {
            "model": "LLaVA-Med v1.5 + LoRA",
            "test_acc": test_acc,
            "test_f1": test_f1
        }
    },
    "training_config": {
        "lora_r": 16,
        "lora_alpha": 32,
        "learning_rate": LEARNING_RATE,
        "batch_size": BATCH_SIZE * GRADIENT_ACCUMULATION,
        "epochs_trained": len(train_losses)
    }
}

with open("experiment_results.json", "w") as f:
```

```
    json.dump(results, f, indent=2)

print("Results saved to experiment_results.json")
```

```
Results saved to experiment_results.json
```

[41]:
```
# Final comparison visualization with all three models
models = ["ResNet50+BiLSTM\n(CNN)", "LLaVA-Med\n(Zero-Shot)", "LLaVA-Med\n(LoRA␣
 ↪Fine-tuned)"]
accuracy = [cnn_results["test_acc"], zs_acc, test_acc]
f1_scores = [cnn_results["test_f1"], zs_f1, test_f1]

x = np.arange(len(models))
width = 0.35

fig, ax = plt.subplots(figsize=(12, 6))
bars1 = ax.bar(x - width/2, accuracy, width, label="Accuracy",␣
 ↪color="steelblue")
bars2 = ax.bar(x + width/2, f1_scores, width, label="Macro-F1", color="coral")

ax.set_ylabel("Score", fontsize=12)
ax.set_title("Complete Model Comparison\nBinary VQA on Chest X-rays (VQA-RAD)",␣
 ↪fontsize=14)
ax.set_xticks(x)
ax.set_xticklabels(models, fontsize=11)
ax.legend(loc="upper right", fontsize=11)
ax.set_ylim(0, 1)

for bar in bars1 + bars2:
    height = bar.get_height()
    ax.annotate(f'{height:.3f}',
                xy=(bar.get_x() + bar.get_width()/2, height),
                xytext=(0, 3), textcoords="offset points",
                ha='center', va='bottom', fontsize=10)

plt.axhline(y=0.5, color='gray', linestyle='--', alpha=0.5)
ax.text(2.5, 0.51, 'Random baseline', fontsize=9, color='gray')

plt.tight_layout()
plt.savefig("complete_model_comparison.png", dpi=150, bbox_inches='tight')
plt.show()
```


Complete Model Comparison
Binary VQA on Chest X-rays (VQA-RAD)
Accuracy
Macro-F1
Score
1.0
0.8
0.6
0.4
0.2
0.0
0.556
0.529
0.524
0.503
0.730
0.723
Random baseline
ResNet50+BiLSTM
(CNN)
LLaVA-Med
(Zero-Shot)
LLaVA-Med
(LoRA Fine-tuned)