**XrayGPT**

**1. Data Preprocessing/Loading (MIMIC-CXR)**

**Paper:**

* MIMIC-CXR images and reports are filtered for quality and paired.
* Clean summaries are generated using GPT-3.5-turbo.
* Data is organized as images + filtered captions.

**Code:**

* Preprocessing: See README-DATASET.md for detailed steps.
* Processed data:

 dataset/mimic/image/ (images), dataset/mimic/filter\_cap.json (annotations).

* Loading:
  + [xraygpt/datasets/datasets/mimic\_dataset.py](https://github.com/gontamar/XrayGPT/blob/main/xraygpt/datasets/datasets/mimic_dataset.py) (MIMIC Dataset)
    - Loads images and captions from the dataset folder.
  + Builder: xraygpt/datasets/builders/image\_text\_pair\_builder.py (uses MIMIC Dataset and annotation paths).

**2. Model Architecture (Vision Encoder, Language Model, Fusion)**

**Paper:**

* Vision encoder: Frozen MedCLIP/EVA ViT.
* Language model: Vicuna (LLaMA-based).
* Q-Former for vision-language fusion.

**Code:**

* Vision encoder: xraygpt/models/eva\_vit.py (EVA ViT).
* Q-Former: xraygpt/models/Qformer.py (BERT-style transformer for modality bridging).
* Language model: xraygpt/models/modeling\_llama.py (LLaMA/Vicuna).
* Integration: xraygpt/models/mini\_gpt4.py (multimodal pipeline).

**3. Training Scripts/Configuration (Hyperparameters, Loss Functions)**

**Paper:**

* Two-stage: pretrain on MIMIC, finetune on Open-i.
* Distributed training, AdamW optimizer, learning rate scheduling.

**Code:**

* Main script: train.py (orchestrates all training stages).
* Config: YAML files in train\_configs/, parsed by xraygpt/common/config.py.
* Optimizer: AdamW, created in xraygpt/runners/runner\_base.py.
* Scheduler: xraygpt/common/optims.py (e.g., LinearWarmupCosineLRScheduler).
* Loss functions: Defined within model/task classes (see xraygpt/models/ and xraygpt/tasks/).

**How to launch (from README):**

bash

torchrun --nproc-per-node NUM\_GPU train.py --cfg-path train\_configs/xraygpt\_mimic\_pretrain.yaml

**4. Evaluation/Metrics**

**Paper:**

* Reports retrieval, similarity metrics, language quality.

**Code:**

* Evaluation loop: xraygpt/runners/runner\_base.py (eval\_epoch, evaluate).
* Metric computation: xraygpt/models/blip2.py (similarity matrices, recall, etc.).
* Logging and checkpointing: in runner\_base.py.

cfg = Config(parse\_args()) - (xraygpt/common/config.py)

|

init\_distributed\_mode(cfg.run\_cfg) - (xraygpt/common/dist\_utils.py)

(run\_cfg) come from common/config.py

|

  setup\_logger() - xraygpt/common/logger.py

|

cfg.pretty\_print() - Pretty print config

(xraygpt/common/config.py)

|

task = tasks.setup\_task(cfg) - xraygpt/tasks/\_\_init\_\_.py, will

instantiate a task class

|

\_\_init\_\_.py

|

Registry. get\_task\_class

|

 datasets = task.build\_datasets(cfg) - xraygpt/datasets/builders/