CLIP Finetune in XTUNE

1. Data Prepare

1.1 Caltech101 Dataset

Xtune support some public dataset. You can refer to https://github.com/jilongW/GenAIComps/blob/main/comps/finetuning/src/integrations/xtune/doc/Prepare_dataset.md for details. Here we use Caltech101 dataset as an example, you can also create and use your own dataset.

If you want to use custom to finetune clip. Please register custom dataset to xtune. Please follow appendix a

1.1.1 Donwload Caltech101 dataset

- Create a folder named `caltech-101/` under `$DATA`.
- Download `101_ObjectCategories.tar.gz` from http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_ObjectCategories.tar.gz and extract the file under `$DATA/caltech-101`.
- Download `split_zhou_Caltech101.json` from this link and put it under `$DATA/caltech-101`.

The directory structure should look like

```
$DATA/
|-- caltech-101/
|   |-- 101_ObjectCategories/
|   | split_zhou_Caltech101.json
```

The split_zhou_Caltech101.json contains how to partition the dataset.

1.1 Regist Caltech101 dataset to llama-factory

make `caltech101.json` in your dataset directory

```
[]
```

then make `dataset_info.json` in your dataset directory

```
{
  "caltech101": {
    "file_name": "caltech101.json"
  }
}
```

The directory structure should look like

```
$DATA/
|-- caltech-101/
|   |-- 101_ObjectCategories/
|   | split_zhou_Caltech101.json
|-- dataset_info.json
|-- caltech101.json
```

2. Install XTUNE

2.1 Install Driver for Arc A770

Please follow https://dgpu-docs.intel.com/driver/installation-rolling.html#ubuntu to install Driver. You can use clinfo -l to check whether driver is installed.

```
$ clinfo -l
Platform #0: Intel(R) OpenCL Graphics
 +-- Device #0: Intel(R) Arc(TM) A770 Graphics
```

2.2 Install XTUNE on native env.

Run install_xtune.sh to prepare component.

```
conda create -n xtune python=3.10 -y
conda activate xtune
git clone https://github.com/opea-project/GenAIComps.git
cd GenAIComps/comps/finetuning/src/integrations/xtune
apt install -y rsync
# open webui as default
bash prepare_xtune.sh
# this way it will not open webui
# bash prepare_xtune.sh false
```

You can see below info when you finished installation.

```
start llamafactory webui
server start successfully
```

You can run bash `prepare_xtune.sh false` to shutdown UI server. And use `bash prepare_xtune.sh` to start UI server again.

2.3 Install xtune on docker

2.3.1 Download Docker Image

You can use upstream docker image. Follow https://hub.docker.com/r/opea/finetuning-xtune

```
docker pull opea/finetuning-xtune
```

Or build docker image with below command:

```
cd ../../../deployment/docker_compose
export DATA="where to find dataset"
docker build -t opea/finetuning-xtune:latest --build-arg
https_proxy=$https_proxy --build-arg http_proxy=$http_proxy --build-arg
HF_TOKEN=$HF_TOKEN --build-arg DATA=$DATA -f
comps/finetuning/src/Dockerfile.xtune .
```

2.3.2 Run Docker with CLI

Suse docker compose with below command:

```
export HF_TOKEN=${your_huggingface_token}
export DATA="where to find dataset"
cd ../../../deployment/docker_compose
docker compose -f compose.yaml up finetuning-xtune -d
```

3. CLIP Finetune Method and Their Configurations

3.1 The core features for clip finetune tool:

Below method can run on Classification task and Image to Text task

Method	Detail Description
Full Finetune	**Enable Angle-Based Selection(base on the weight angle to determine which layer to update)**
Partial Finetuning - bias	**Allow users to customize which layers participate in training and which ones do not**
Prompt Tuning	**adding prompt embedding layer at the head of model or at the inputs of every layer and only train these layers**
Adapter Tuning	**adding adapter network at the end of encoder and only train this network**
Training free - Tip Adapter	**1. finetune CLIP model without any training or with few epochs learning** **2. Added fixed cache size to reduce memory and enable experience sharing across different datasets**

3.2 Config features for clip finetune tool in command line: {id=3_2}

3.2.1 Basic yaml for vit_b16.yaml

see this in src/llamafactory/clip_finetune/configs/clip_finetune/vit_b16.yaml

```
DATALOADER:
  TRAIN_X:
    BATCH_SIZE: 32              # Train batch size
  TEST:
    BATCH_SIZE: 100             # Test batch size
  NUM_WORKERS: 4

INPUT:
  SIZE: (224, 224)              # resize image to this size
  INTERPOLATION: "bicubic"
  PIXEL_MEAN: [0.48145466, 0.4578275, 0.40821073]
  PIXEL_STD: [0.26862954, 0.26130258, 0.27577711]
  TRANSFORMS: ["random_resized_crop", "random_flip", "normalize"]

OPTIM:
  NAME: "sgd"
  LR: 0.02
  MAX_EPOCH: 50
  LR_SCHEDULER: "cosine"
  WARMUP_EPOCH: 1
  WARMUP_TYPE: "constant"
  WARMUP_CONS_LR: 1e-5

TRAIN:
  PRINT_FREQ: 1             # print acc after $PRINT_FREQ iteration

MODEL:
  BACKBONE:
    NAME: "ViT-B/16"
```

3.2.2 Angle-Based Selection for full-finetune

Add below line in src/llamafactory/clip_finetune/configs/clip_finetune/vit_b16.yaml

```
MODEL:
  ABS: True
  ABS_TOP: True                         # True: select top ABS_KEEP layer
False: select bottom ABS_KEEP layer
  ABS_GROUP: True                       # True: select top ABS_KEEP layer in
each group False: select bottom ABS_KEEP layer
  ABS_GROUP_NAME: ["k_proj", "v_proj", "q_proj"]     # How to divide layer
into GTOUP, this means divide layers into 4 group. Each layer has k_proj in
its name will into group 0, v_proj into group1, q_proj into group 2, other
into group 3
```

```
  ABS_KEEP: 5                       # keep layer number
  BACKBONE:
    NAME: "ViT-B/16"
```

3.2.3 customize trained layer for partial-finetune

Add below line in src/llamafactory/clip_finetune/configs/clip_finetune/vit_b16.yaml

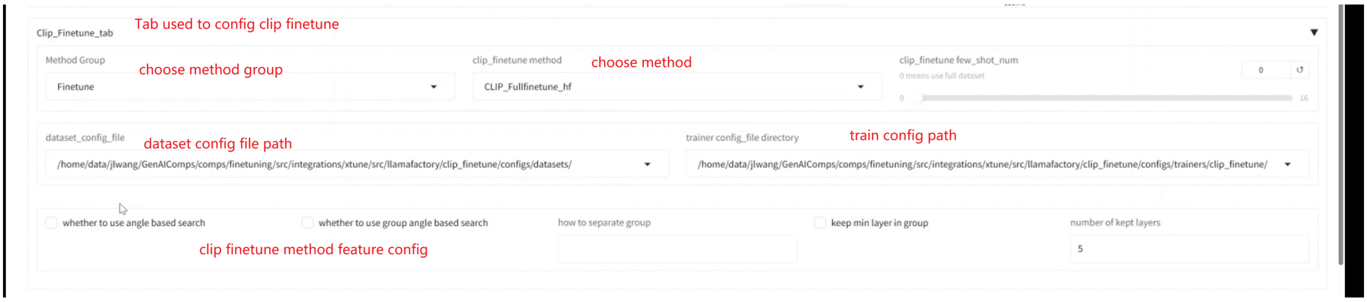
```
MODEL:
  BACKBONE:
    NAME: "ViT-B/16"
BIAS:
  BIAS_TERMS: ["layer_norm", "layernorm"]   # which layer you want to train
  BIAS_TERMS_EXCLUDE: ["layernorm"]         # which layer you don't want to
train
```

3.2.4 fixed cache size for tip-adapter

Add below line in src/llamafactory/clip_finetune/configs/clip_finetune/vit_b16.yaml

```
TRAINER:
  TIP:
    LOAD_CACHE: True                  # whether to use cache data trained
with tip-adapter before
    beta: 1.0                         # hyper param in origin paper
    alpha: 3.0                        # hyper param in origin paper
    AUGMENT_EPOCH: 10                 # train cache epoch
    search_best: True                 # whether to search the best beta
and alpha
    NEW: False                        # whether to use fixed cache size.
True: all dataset cache will merge into one tensor [100, hidden_size]
False: each dataset will has it's own cache [num_dataset * 100,
hidden_size]
    NEW_DATASET: False                # Whether to train this dataset
from scratch
```

3.3 Config features for clip finetune tool in UI:



Clip_Finetune_tab

Tab used to config clip finetune

Method Group: Finetune

choose method group

clip_finetune method: CLIP_Fullfinetune_hf

choose method

clip_finetune few_shot_num: 0
0 means use full dataset

dataset_config_file: /home/data/jlwang/GenAIComps/comps/finetuning/src/integrations/xtune/src/llamafactory/clip_finetune/configs/datasets/

dataset config file path

trainer config_file directory: /home/data/jlwang/GenAIComps/comps/finetuning/src/integrations/xtune/src/llamafactory/clip_finetune/configs/trainers/clip_finetune/

train config path

□ whether to use angle based search

□ whether to use group angle based search

how to separate group:

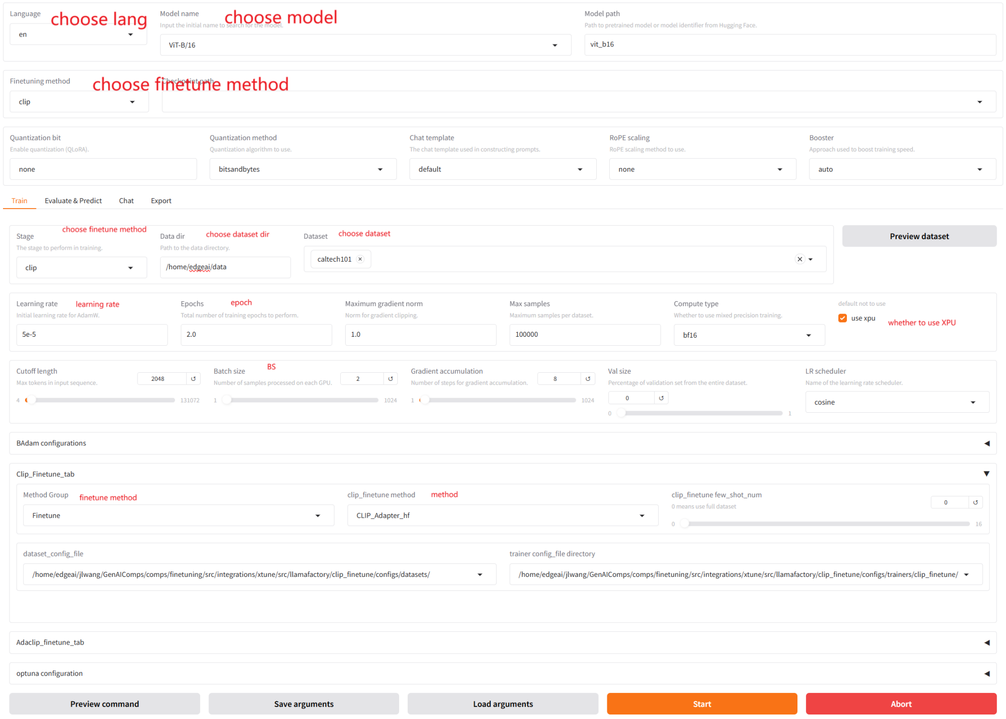
□ keep min layer in group

number of kept layers: 5

clip finetune method feature config

4. Finetune Clip with GUI

4.1 How to finetune



Language: en

Model name: ViT-B/16 (Input the initial name to search for the model.)

Model path: vit_b16 (Path to pretrained model or model identifier from Hugging Face.)

Finetuning method: clip

Checkpoint path:

Quantization bit: none (Enable quantization (QLoRA).)

Quantization method: bitsandbytes (Quantization algorithm to use.)

Chat template: default (The chat template used in constructing prompts.)

RoPE scaling: none (RoPE scaling method to use.)

Booster: auto (Approach used to boost training speed.)

Stage: clip (The stage to perform in training.)

Data dir: /home/edgeai/data (Path to the data directory.)

Dataset: caltech101

Learning rate: 5e-5 (Initial learning rate for AdamW.)

Epochs: 2.0 (Total number of training epochs to perform.)

Maximum gradient norm: 1.0 (Norm for gradient clipping.)

Max samples: 100000 (Maximum samples per dataset.)

Compute type: bf16 (Whether to use mixed precision training.)

default not to use

☑ use xpu

Cutoff length: 2048 (Max tokens in input sequence.)

Batch size: 2 (Number of samples processed on each GPU.)

Gradient accumulation: 8 (Number of steps for gradient accumulation.)

Val size: 0 (Percentage of validation set from the entire dataset.)

LR scheduler: cosine (Name of the learning rate scheduler.)

Method Group: Finetune

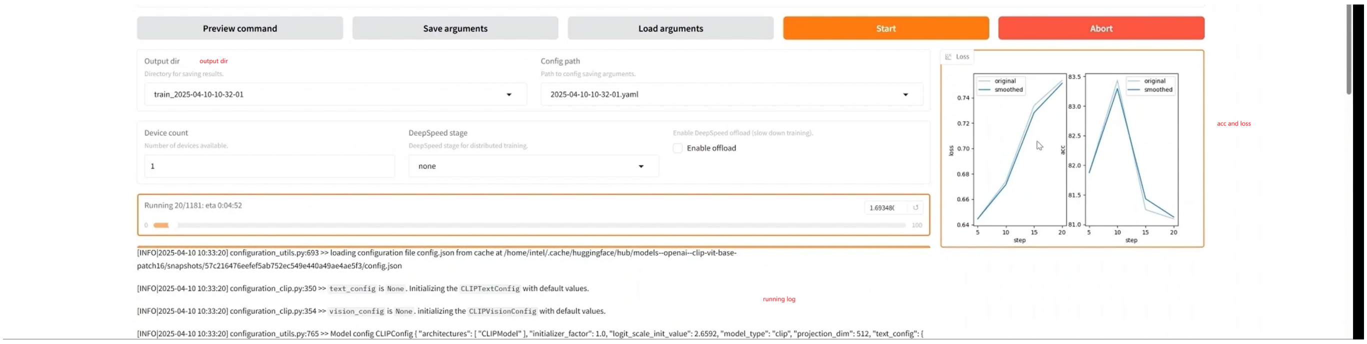
clip_finetune method: CLIP_Adapter_hf

clip_finetune few_shot_num: 0 (0 means use full dataset)

dataset_config_file: /home/edgeai/jlwang/GenAIComps/comps/finetuning/src/integrations/xtune/src/llamafactory/clip_finetune/configs/datasets/

trainer config_file directory: /home/edgeai/jlwang/GenAIComps/comps/finetuning/src/integrations/xtune/src/llamafactory/clip_finetune/configs/trainers/clip_finetune/

4.2 Check the result



Output dir: train_2025-04-10-10-32-01 (Directory for saving results.)

Config path: 2025-04-10-10-32-01.yaml (Path to config saving arguments.)

Device count: 1 (Number of devices available.)

DeepSpeed stage: none (DeepSpeed stage for distributed training.)

Enable DeepSpeed offload (slow down training).

□ Enable offload

Running 20/1181: eta 0:04:52

[INFO|2025-04-10 10:33:20] configuration_utils.py:693 >> loading configuration file config.json from cache at /home/intel/.cache/huggingface/hub/models--openai--clip-vit-base-patch16/snapshots/57c216476eefef5ab752ec549e440a49ae4ae5f3/config.json

[INFO|2025-04-10 10:33:20] configuration_clip.py:350 >> `text_config` is `None`. Initializing the `CLIPTextConfig` with default values.

[INFO|2025-04-10 10:33:20] configuration_clip.py:354 >> `vision_config` is `None`. initializing the `CLIPVisionConfig` with default values.

[INFO|2025-04-10 10:33:20] configuration_utils.py:765 >> Model config CLIPConfig { "architectures": ["CLIPModel"], "initializer_factor": 1.0, "logit_scale_init_value": 2.6592, "model_type": "clip", "projection_dim": 512, "text_config": {

4.3 Checkpoint and Log

see in GenAIComps/comps/finetuning/src/integrations/xtune/saves/ViT-B/16/clip/${Output dir}

```
$ ls GenAIComps/comps/finetuning/src/integrations/xtune/saves/ViT-B/16/clip/train_2025-05-15-17-18-28
| clip_adapter_hf                          # store checkpoint
| -- checkpoint
| -- model.pth.tar-2
| llamaboard_config.yaml
| tensorboard
| trainer_log.jsonl                        # training log
| training_args.yaml
```

4.4 Load finetuned model

```
cd GenAIComps/comps/finetuning/src/integrations/xtune/src/llamafactory/clip_finetune
```

run below example to load the model

```
import argparse
import gc
import os
import shutil
import sys

import optuna
import torch
import torch.distributed as dist
import torch.nn.parallel
from dassl.config import get_cfg_default
from dassl.engine import build_trainer
from dassl.utils import collect_env_info, set_random_seed, setup_logger
from datasets import *
from optuna.trial import TrialState
from trainers import *
from train import *
import yaml



if __name__ == "__main__":
    cfg = get_cfg_default()
    extend_cfg(cfg)
    cfg.TRAINER.COOP.XPU = True
    cache_dir = "/home/edgeai/jlwang/GenAIComps/comps/finetuning/src/integrations/xtune/saves/ViT-B/16/clip/train_2025-05-15-17-18-28"
    with open(cache_dir + '/training_args.yaml', 'r') as file:
        data = yaml.safe_load(file)

    cfg.merge_from_file(data["dataset_config_file"] + data["dataset"] + ".yaml")
    cfg.merge_from_file(data["config_file"] + data["model_name_or_path"] + ".yaml")
    cfg.TRAINER.NAME = data["trainer"]
    cfg.DATASET.ROOT = data["root"]
    print(cfg)
    trainer = build_trainer(cfg)


    trainer.load_model(cache_dir , epoch=2)
    print(trainer.model)
```

5. Finetune Clip with SHELL instead of GUI

5.1 How to finetune

You can use our exists finetune script

```
cd GenAIComps/comps/finetuning/src/integrations/xtune/src/llamafactory/clip_finetune
bash scripts/clip_finetune/clip_adapter_hf.sh caltech101 vit_b16 0 xpu >> test_log.txt
```

Bolow are scripts details

```
#!/bin/bash

# Copyright (C) 2025 Intel Corporation
# SPDX-License-Identifier: Apache-2.0

# custom config

TRAINER=CLIP_Adapter_hf       # Config train method
DATASET=$1                    # Config dataset
CFG=$2                        # Config finetune param, see sec 3.2
ACCEPT=$3                     # Config how much steop to print acc
DEVICE=$4                     # Config which device you want to use
USE_OPTUNA=$5                 # Config whether to use optuna
device=0
if [ -z $ACCEPT ]; then
    ACCEPT=0
fi
if [ -z $DEVICE ]; then
    DEVICE="cuda"
    device=0
fi
if [ -z $USE_OPTUNA ]; then
    USE_OPTUNA=0
fi
if [ $DEVICE = "XPU" ]; then
    device=1
    ZE_AFFINITY_MASK=0 python train.py \
    --root ${DATA} \
    --trainer ${TRAINER} \
    --dataset-config-file configs/datasets/${DATASET}.yaml \
    --config-file configs/trainers/clip_finetune/${CFG}.yaml \
    --output-dir output/${TRAINER}/${CFG}/${DATASET} \
    --xpu $device \
    --seed 123 \
    --use-optuna $USE_OPTUNA \
    TRAINER.COOP.ACC $ACCEPT \
    TRAINER.COOP.N_CTX 16 \
    TRAINER.COOP.CSC True \
    TRAINER.COOP.CLASS_TOKEN_POSITION end \
    DATASET.NUM_SHOTS 0
else
    CUDA_VISIBLE_DEVICES=0 python train.py \
    --root ${DATA} \
    --trainer ${TRAINER} \
    --dataset-config-file configs/datasets/${DATASET}.yaml \
    --config-file configs/trainers/clip_finetune/${CFG}.yaml \
    --output-dir output/${TRAINER}/${CFG}/${DATASET} \
    --xpu $device \
    --seed 123 \
    --use-optuna $USE_OPTUNA \
    --eval-only \
    TRAINER.COOP.ACC $ACCEPT \
    TRAINER.COOP.N_CTX 16 \
    TRAINER.COOP.CSC True \
    TRAINER.COOP.CLASS_TOKEN_POSITION end \
    DATASET.NUM_SHOTS 16
fi
```

5.2 Check the result

You can see the result in terminal

```
$ cat test_log.txt
....
=> result
* total: 2,465
* correct: 1,635
* accuracy: 66.3%
* error: 33.7%
* macro_f1: 52.3%
=============================finish_test====================
Elapsed: 1:55:05
```

Appendix

Appendix a: Custom Dataset {id=appendix_a}

You need to wite a python file to tell clip how to partition the dataset and put it under `$PATH/comps/finetuning/src/integrations/xtune/src/llamafactory/clip_finetune/datasets`. You can refer to these python file as example.https://github.com/jilongW/Dassl.pytorch/tree/master/clip/datasets After this, please restart xtune.

We use caltech101 as excample:

```
# Copyright (C) 2025 Intel Corporation
# SPDX-License-Identifier: Apache-2.0

import os
import pickle
from collections import OrderedDict

from dassl.data.datasets import DATASET_REGISTRY, DatasetBase, Datum
from dassl.utils import listdir_nohidden, mkdir_if_missing

from .oxford_pets import OxfordPets


@DATASET_REGISTRY.register()
class MiniImageNet(DatasetBase):                    # dataset class

    dataset_dir = "mini-imagenet"                   # dataset folder name

    def __init__(self, cfg):
        root = os.path.abspath(os.path.expanduser(cfg.DATASET.ROOT))
        self.dataset_dir = os.path.join(root, self.dataset_dir)
        self.image_dir = self.dataset_dir
        self.preprocessed = os.path.join(self.dataset_dir,
"preprocessed.pkl")
        self.split_fewshot_dir = os.path.join(self.dataset_dir,
"split_fewshot")
        mkdir_if_missing(self.split_fewshot_dir)

        if os.path.exists(self.preprocessed):       # read from cache
            with open(self.preprocessed, "rb") as f:
                preprocessed = pickle.load(f)
                train = preprocessed["train"]
                test = preprocessed["test"]
                val = preprocessed["val"]
        else:                                       # the first time to run
            # classnames.txt like this "n01440764 tench", the folder name
and lable. It means all picture in this folder are belong to this lable.
            text_file = os.path.join(self.dataset_dir, "classnames.txt")
            classnames = self.read_classnames(text_file)
            # register all train picture
            train = self.read_data(classnames, "train")
            # print(train)
            # Follow standard practice to perform evaluation on the val set
            # Also used as the val set (so evaluate the last-step model)
            test = self.read_data(classnames, "test")
            # print(test)
            val = self.read_data(classnames, "val")
            # print(val)

            # save cache
            preprocessed = {"train": train, "test": test, "val": val}
            with open(self.preprocessed, "wb") as f:
                pickle.dump(preprocessed, f,
protocol=pickle.HIGHEST_PROTOCOL)

        num_shots = cfg.DATASET.NUM_SHOTS
        if num_shots >= 1:
            seed = cfg.SEED
            preprocessed = os.path.join(self.split_fewshot_dir,
f"shot_{num_shots}-seed_{seed}.pkl")

            if os.path.exists(preprocessed):
                print(f"Loading preprocessed few-shot data from
{preprocessed}")
                with open(preprocessed, "rb") as file:
                    data = pickle.load(file)
                    train, val = data["train"], data["val"]
            else:
                train = self.generate_fewshot_dataset(train,
num_shots=num_shots)
                val = self.generate_fewshot_dataset(val,
num_shots=min(num_shots, 4))
                data = {"train": train, "val": val}
                print(f"Saving preprocessed few-shot data to
{preprocessed}")
                with open(preprocessed, "wb") as file:
                    pickle.dump(data, file,
protocol=pickle.HIGHEST_PROTOCOL)

        subsample = cfg.DATASET.SUBSAMPLE_CLASSES
        train, val, test = OxfordPets.subsample_classes(train, val, test,
subsample=subsample)

        super().__init__(train_x=train, val=val, test=test)

    @staticmethod
    def read_classnames(text_file):
        """Return a dictionary containing
        key-value pairs of <folder name>: <class name>."""
        classnames = OrderedDict()
        with open(text_file, "r") as f:
            lines = f.readlines()
            for line in lines:
                line = line.strip().split(" ")
                folder = line[0]
                classname = " ".join(line[1:])
                classnames[folder] = classname
        return classnames

    def read_data(self, classnames, split_dir):
        split_dir = os.path.join(self.image_dir, split_dir)
        folders = sorted(f.name for f in os.scandir(split_dir) if
f.is_dir())
        items = []

        for label, folder in enumerate(folders):
            imnames = listdir_nohidden(os.path.join(split_dir, folder))
            classname = classnames[folder]
            for imname in imnames:
                impath = os.path.join(split_dir, folder, imname)
                item = Datum(impath=impath, label=label,
classname=classname)
                items.append(item)

        return items
```