lorenzo-reid-baseline document

folder, file, class, function

playreid:

✧ data

data.datasets

This folder contains classes of different datasets. Different classes inherit class ImageDataset in playreid.data.dataset.bases.py .
``__getitem__ `` returns an image given index. It will return ``img``, ``pid``, ``camid`` and ``img_path`` where ``img`` has shape (channel, height, width). As a result, data in each batch has shape (batch_size, channel, height, width).
Specially, market1501, cuhk03, dukemtmcreid, msmt17 classes and datasets have been modified to the uniform format which is convenient to train four datasets jointly.

data.samplers

This folder contains classes of different samplers. Use NaiveIdentitySampler by default.

data.transforms

This folder is used to do transforms when training and inferring.

By default, we use the following settings:
Train: Resize, RandomHorizontalFlip, Pad, RandomCrop, ToTensor, Normalize and RandomErasing.
Infer: Resize, ToTensor, Normalize.

data.build

Build train&test dataloader and define batch_collator.

data.common

Define CommDataset to get item at each mini-batch.

· CommDataset

Basic image Person ReID Dataset

data.data_utils

· DataloaderX

Use BackgroundGenerator to accelerate data i/o:
#based on http://stackoverflow.com/questions/7323664/python-generator-pre-fetch
This is a single-function package that transforms arbitrary generator into a background-thead generator that prefetches several batches of data in a parallel background thead.

✧ engine

engine.launch

(The ReIDTrainer may be added in the future.)

· def launch():

# Modified from the file of fast-reid
Launch multi-gpu or distributed training.
This function must be called on all machines involved in the training.
It will spawn child processes (defined by num_gpus_per_machine) on each machine. Concretely, this function will start multi process by torch.multiprocessing.spawn() if your world_size > 1.

· def _distributed_worker():

This function initializes the DDP and synchronizes multi process.

You can also add the following code to print the rank of current process after initialization:

```
print(f"This is engine.launch, local_rank is: {local_rank}")
```

✧ evaluation

evaluation.evaluator

# Copyright (c) Facebook, Inc. and its affiliates. All Rights Reserved
The function inference_on_dataset() is used to run model on the data_loader and evaluate the metrics with evaluator.

evaluation.query_expansion

# based on
#
https://github.com/PyRetri/PyRetri/blob/master/pyretri/index/re_ranker/re_ranker_impl/query_expansion.py
The function aqe() is used to combine the retrieved topk nearest neighbors with the original query and doing another retrieval. while testing with cfg.TEST.AQE.ENABLE=True.

evaluation.rank

# credits:
https://github.com/KaiyangZhou/deep-person-reid/blob/master/torchreid/metrics/rank.py
Use python or cython to evaluate CMC rank by the distance matrix. Default is Cython which accelerates CMC calculation and it's recommended to use.

evaluation.reid_evaluation

The function process() of class ReidEvaluator is used to record feature vectors. The function evaluate() is used to calculate the distance matrix of query&gallery and evaluate Rank1/5/10, mAP, mINP and metrics((Rank1+mAP) / 2) according to CMC generated from evaluation.rank.

evaluation.rerank

# based on:
# https://github.com/zhunzhong07/person-re-ranking
This is to do re-ranking.

evaluation.testing

# Copyright (c) Facebook, Inc. and its affiliates. All Rights Reserved
Visualize the results in tabular form.

✧ layers

layers.activation

Define some activate functions: Mish, Swish, MemoryEfficientSwish and GELU.

layers.any_softmax

Define softmax: Linear, CosSoftmax, ArcSoftmax and CircleSoftmax.

layers.batch_norm

Define BN: BatchNorm, SyncBatchNorm, IBN, GhostBatchNorm and FrozenBatchNorm.

layers.non_local

Define the non_local module.

layers.se_layer

Define the SE module.

layers.pooling

Define pooling: Identity, Flatten, GlobalAvgPool, GlobalMaxPool, GeneralizedMeanPooling, GeneralizedMeanPoolingP, FastGlobalAvgPool, AdaptiveAvgMaxPool and ClipGlobalAvgPool.

layers.splat

Define SplAtConv2d used in ResNeSt.

✧ modeling

modeling.backbones

This folder contains different backbones like: mobilenet, resnet, resnest, shufflenet, etc.

modeling.heads

This folder contains different heads adapt to ReID task.

modeling.losses

This folder contains different losses like: circle_loss, cross_entroy_loss, focal_loss and triplet_loss.

modeling.losses

baseline in this folder registers the ReID model with responding backbone and head.

✧ utils

utils.collect_env

# Copyright (c) Facebook, Inc. and its affiliates. All Rights Reserved.

- **class Checkpointer:**

A checkpointer that can save/load model as well as extra checkpointable objects. More details are in the code annotations.
You can use the class directly like:

```
model_path = "./logs/model.pth"
output_dir = "./logs"
file_name = "my_model"
model = MyNet()
Checkpointer(model).load(model_path)
Checkpointer(model, output_dir).save(file_name)
```

Or you can instantiate the class like:

```
model_path = "./logs/model.pth"
output_dir = "./logs"
file_name = "my_model"
model = MyNet()
optimizer = build_optimizer()  # build torch.optim
scheduler = build_lr_scheduler() # build scheduler_dict
checkpointer = Checkpointer(model, output_dir, save_to_disk=True, optimi
zer=optimizer, **scheduler)
Checkpointer(model).load(model_path)
checkpointer.save(file_name)
```

- **class PeriodCheckpointer:**

Save checkpoints periodically. When ``.step(iteration)`` is called, it will execute ``checkpointer.save`` on the given checkpointer, if iteration is a multiple of period or if ``max_iter`` is reached.

utils.collect_env

#based on
#https://github.com/facebookresearch/detectron2/blob/master/detectron2/utils/collect_env.py

· def collect_env_info():

This function is used to collect environment information such as system version, python version, pytorch version, etc.

utils.comm

This file contains primitives for multi-gpu communication which is useful when doing distributed training. Concretely, there are several functions to get world size, get (local) rank, synchronize multi process, etc. A torch process group which only includes processes that on the same machine as the current process. This variable is set when processes are spawned by `launch()` in "engine/launch.py".

utils.compute_dist

# Modified from:
https://github.com/open-mmlab/OpenUnReID/blob/66bb2ae0b00575b80fbe8915f4d4f4739cc21206/openunreid/core/utils/compute_dist.py
Compute cosine/euclidean/jaccard distance between two feature embeddings.

utils.env

# Copyright (c) Facebook, Inc. and its affiliates. All Rights Reserved
The function seed_all_rng() is used to set the random seed for the RNG in torch, numpy and python.

utils.events

# Copyright (c) Facebook, Inc. and its affiliates.
This file is used to record training details.

· class EventStorage:

The user-facing class that provides metric storage functionalities.

· class JSONWriter:

Write scalars to a json file. It saves scalars as one json per line (instead of a big json) for easy parsing.

· class CommonMetricPrinter:

Print **common** metrics to the terminal, including iteration time, ETA, memory, all losses and the learning rate. It also applies smoothing using a window of 20 elements. It's meant to print common metrics in common ways.

- **class TensorboardWriter:**

Write all scalars to a tensorboard file.

utils.file_io

# Copyright (c) Facebook, Inc. and its affiliates. All Rights Reserved

- **class PathManager:**

A class for users to open generic paths or translate generic paths to file names. You can use the class to do file i/o operation: open, copy, get_local_path, exits, isfile, isdir, etc. Details can be found in corresponding function.

For example:

```
output_dir = "./log"
file_path = "./configs/model.yml"
dst_path = "./configs/model_copy.yml"
PathManager.mkdirs(output_dir)
PathManager.copy(file_path, dst_path)
PathManager.open(file_path, "r")
PathManager.isdir(output_dir)
PathManager.isfile(file_path)
```

utils.logger

# Copyright (c) Facebook, Inc. and its affiliates. All Rights Reserved

- **def setup_logger():**

This function sets up the logger in all processes.

utils.measure_model_sparsity

This file calculates the sparsity of global model and each layer.

utils.register

# Copyright (c) Facebook, Inc. and its affiliates. All Rights Reserved

- **class Registry:**

The registry that provides name -> object mapping.

To create a registry (e.g. a backbone registry):

```
BACKBONE_REGISTRY = Registry('BACKBONE')
```

To register an object:

```python
@BACKBONE_REGISTRY.register()
class MyBackbone():
```

Or:

```python
BACKBONE_REGISTRY.register(MyBackbone)
```