git clone <https://github.com/JDAI-CV/fast-reid.git>

Get inside the folder

Make it work following

<https://github.com/JDAI-CV/fast-reid/blob/master/INSTALL.md>

Generate a prototype of setup.py that will be enough:

from setuptools import setup, find\_packages

def find\_version():

    version\_file = "fastreid/\_\_init\_\_.py"

    with open(version\_file, "r") as f:

        exec(compile(f.read(), version\_file, "exec"))

    return locals()["\_\_version\_\_"]

setup(

    name="fastreid",

    version=find\_version(),

    description="FastReID: A Pytorch Toolbox for General Instance Re-identification",

    author="He, Lingxiao and Liao, Xingyu and Liu, Wu and Liu, Xinchen and Cheng, Peng and Mei, Tao",

    license="Apache License 2.0",

    url="https://github.com/JDAI-CV/fast-reid",

    packages=find\_packages(),

    keywords=["Re-Identification", "Deep Learning", "Computer Vision"],

)

Install with:

pip install -e .

“import fastreid” will barely work, it won’t import modules or subpackages, just \_\_init\_\_.py. It is better imports with the form “from fastreid.XXXX import YYYY”

1. In case of:

ImportError: cannot import name 'Mapping' from 'collections' (/usr/lib/python3.10/collections/\_\_init\_\_.py)

Edit fastreid/evaluation/testing.py

Line 5 from

from collections import Mapping, OrderedDict

To

from collections.abc import Mapping

from collections import OrderedDict

And edit fastreid/data/build.py

Line 19 from

from collections import Mapping

To

from collections.abc import Mapping

1. In order to train with it, make correct yaml configs or put datasets in the installation path. And make it exist on the lunching directory (ln -s)

Edit fastreid/data/transforms/build.py lines 66, 70 & 96 from:

interpolation=3

to

interpolation=T.InterpolationMode.BICUBIC

Edit fastreid/data/transforms/build.py line 81 from:

resample=False

to

interpolation=T.InterpolationMode.NEAREST

Edit fastreid/data/transforms/build.py line 82 from:

fillcolor=0

to

fill=0

1. Trick to customize the training as I want:

EDIT fastreid/config/defaults.py (Add wherever)

# random rotation

\_C.INPUT.ROTATION = CN({"ENABLED": False})

\_C.INPUT.ROTATION.DEGREES = 0

#\_C.INPUT.ROTATION.PROB = 0.5

EDIT fastreid/data/transforms/build.py (Overwrite all)

# encoding: utf-8

"""

@author:  liaoxingyu

@contact: sherlockliao01@gmail.com

"""

import torchvision.transforms as T

from .transforms import \*

from .autoaugment import AutoAugment

def build\_transforms(cfg, is\_train=True):

    res = []

    if is\_train:

        size\_train = cfg.INPUT.SIZE\_TRAIN

        # augmix augmentation

        do\_augmix = cfg.INPUT.AUGMIX.ENABLED

        augmix\_prob = cfg.INPUT.AUGMIX.PROB

        # horizontal filp

        do\_flip = cfg.INPUT.FLIP.ENABLED

        flip\_prob = cfg.INPUT.FLIP.PROB

        # color jitter

        do\_cj = cfg.INPUT.CJ.ENABLED

        cj\_prob = cfg.INPUT.CJ.PROB

        cj\_brightness = cfg.INPUT.CJ.BRIGHTNESS

        cj\_contrast = cfg.INPUT.CJ.CONTRAST

        cj\_saturation = cfg.INPUT.CJ.SATURATION

        cj\_hue = cfg.INPUT.CJ.HUE

        # random rotation

        do\_rot = cfg.INPUT.ROTATION.ENABLED

        #rot\_prob = cfg.INPUT.ROTATION.PROB

        rot\_degrees = cfg.INPUT.ROTATION.DEGREES

        if size\_train[0] > 0:

            res.append(T.Resize(size\_train[0] if len(size\_train) == 1 else size\_train, interpolation=T.InterpolationMode.BICUBIC))

        if do\_flip:

            res.append(T.RandomHorizontalFlip(p=flip\_prob))

        if do\_rot:

            #res.append(T.RandomApply(T.RandomRotation(degrees=rot\_degrees), p=rot\_prob))

            res.append(T.RandomRotation(degrees=rot\_degrees))

        if do\_cj:

            res.append(T.RandomApply([T.ColorJitter(cj\_brightness, cj\_contrast, cj\_saturation, cj\_hue)], p=cj\_prob))

        if do\_augmix:

            res.append(AugMix(prob=augmix\_prob))

        res.append(ToTensor())

    else:

        size\_test = cfg.INPUT.SIZE\_TEST

        if size\_test[0] > 0:

            res.append(T.Resize(size\_test[0] if len(size\_test) == 1 else size\_test, interpolation=T.InterpolationMode.BICUBIC))

        res.append(ToTensor())

    return T.Compose(res)