change source code

```
import os
import tensorflow as tf
tf.reset_default_graph()
from ops_c import discriminator, generator_gatedcnn
from utils import l1_loss, l2_loss, cross_entropy_loss
from datetime import datetime
# -*- coding: utf-8 -*-
# /usr/bin/python2

from __future__ import print_function

import argparse
import os

import tensorflow as tf
from tensorpack.callbacks.saver import ModelSaver
from tensorpack.input_source.input_source import QueueInput
from tensorpack.tfutils.sessinit import ChainInit
from tensorpack.tfutils.sessinit import SaverRestore
from tensorpack.train.interface import TrainConfig
from tensorpack.train.interface import launch_train_with_config
from tensorpack.train.trainers import SyncMultiGPUTrainerReplicated
from tensorpack.utils import logger

from data_load import Net2DataFlow
from hparam import hparam as hp
from models import Net2
from utils import remove_all_files


def train(args, logdir1, logdir2):
    # model
    model = Net2()

    # dataflow
    df = Net2DataFlow(hp.train2.data_path, hp.train2.batch_size)

    # set logger for event and model saver
    logger.set_logger_dir(logdir2)

    # session_conf = tf.ConfigProto(
    #     gpu_options=tf.GPUOptions(
    #         allow_growth=True,
    #         per_process_gpu_memory_fraction=0.6,
    #     ),
    # )

    session_inits = []
    ckpt2 = '{}/{}'.format(logdir2, args.ckpt) if args.ckpt else tf.train.latest_checkpoint(logdir2)
    if ckpt2:
        session_inits.append(SaverRestore(ckpt2))
    ckpt1 = tf.train.latest_checkpoint(logdir1)
    if ckpt1:
        session_inits.append(SaverRestore(ckpt1, ignore=['global_step']))
    train_conf = TrainConfig(
        model=model,
        data=QueueInput(df(n_prefetch=1000, n_thread=4)),
        callbacks=[
            # TODO save on prefix net2
            ModelSaver(checkpoint_dir=logdir2),
            # ConvertCallback(logdir2, hp.train2.test_per_epoch),
        ],
        max_epoch=hp.train2.num_epochs,
        steps_per_epoch=hp.train2.steps_per_epoch,
        session_init=ChainInit(session_inits)
    )
    if args.gpu:
        os.environ['CUDA_VISIBLE_DEVICES'] = args.gpu
        train_conf.nr_tower = len(args.gpu.split(','))

    trainer = SyncMultiGPUTrainerReplicated(hp.train2.num_gpu)

    launch_train_with_config(train_conf, trainer=trainer)


# def get_cyclic_lr(step):
#     lr_margin = hp.train2.lr_cyclic_margin * math.sin(2. * math.pi / hp.train2.lr_cyclic_steps * step)
#     lr = hp.train2.lr + lr_margin
#     return lr


def get_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument('case1', type=str, help='experiment case name of train1')
    parser.add_argument('case2', type=str, help='experiment case name of train2')
    parser.add_argument('-ckpt', help='checkpoint to load model.')
    parser.add_argument('-gpu', help='comma separated list of GPU(s) to use.')
    parser.add_argument('-r', action='store_true', help='start training from the beginning.')
    arguments = parser.parse_args()
    return arguments


if __name__ == '__main__':
    args = get_arguments()
    hp.set_hparam_yaml(args.case2)
    logdir_train1 = '{}/{}/train1'.format(hp.logdir_path, args.case1)
    logdir_train2 = '{}/{}/train2'.format(hp.logdir_path, args.case2)

    if args.r:
        remove_all_files(logdir_train2)

    print('case1: {}, case2: {}, logdir1: {}, logdir2: {}'.format(args.case1, args.case2, logdir_train1, logdir_train2))

    train(args, logdir1=logdir_train1, logdir2=logdir_train2)

    print("Done")
```