Propagation Regularizer for Semi-supervised Learning with Extremely Scarce Labeled Samples

This document is the instruction of code for “**Propagation Regularizer for Semi-supervised Learning with Extremely Scarce Labeled Samples**” (Paper ID 11752).

Our code is based on the official FixMatch (https://github.com/google-research/fixmatch).

Firstly, clone the official FixMatch project, and place “fixmatch_reg.py” and “uda_reg.py” to the root directory of the project.

```
git clone https://github.com/google-research/fixmatch.git
mv fixmatch_reg.py uda_reg.py ./fixmatch/
cd fixmatch
```

"fixmatch_reg.py" and “uda_reg.py” are codes with our proposed propagation regularizer.

Default instruction follows the official Fixmatch.

Setup

Install dependencies

```
sudo apt install python3-dev python3-virtualenv python3-tk imagemagick
virtualenv -p python3 --system-site-packages env3
. env3/bin/activate
pip install -r requirements.txt
```

Install datasets

```
export ML_DATA="./dataset"
export PYTHONPATH=$PYTHONPATH:"./"      # project root

# Download datasets
CUDA_VISIBLE_DEVICES= ./scripts/create_datasets.py
cp $ML_DATA/svhn-test.tfrecord $ML_DATA/svhn_noextra-test.tfrecord

# Create unlabeled datasets
CUDA_VISIBLE_DEVICES= scripts/create_unlabeled.py $ML_DATA/SSL2/svhn $ML_DATA/svhn-train.tfrecord $ML_DATA/svhn-extra.tfrecord &
CUDA_VISIBLE_DEVICES= scripts/create_unlabeled.py $ML_DATA/SSL2/cifar10 $ML_DATA/cifar10-train.tfrecord &
CUDA_VISIBLE_DEVICES= scripts/create_unlabeled.py $ML_DATA/SSL2/cifar100 $ML_DATA/cifar100-train.tfrecord &
wait

# Create semi-supervised subsets
for seed in 1 2 3 4 5; do
    for size in 10 20 40 100 250; do
        CUDA_VISIBLE_DEVICES= scripts/create_split.py --seed=$seed --size=$size $ML_DATA/SSL2/svhn $ML_DATA/svhn-train.tfrecord $ML_DATA/svhn-extra.tfrecord &
        CUDA_VISIBLE_DEVICES= scripts/create_split.py --seed=$seed --size=$size $ML_DATA/SSL2/cifar10 $ML_DATA/cifar10-train.tfrecord &
    done
    for size in 100 200 400 1000 2500; do
        CUDA_VISIBLE_DEVICES= scripts/create_split.py --seed=$seed --size=$size $ML_DATA/SSL2/cifar100 $ML_DATA/cifar100-train.tfrecord &
    done
    wait
done
```

Running

Setup

All commands must be ran from the project root. The following environment variables must be defined:

```
export ML_DATA="./dataset"
export PYTHONPATH=$PYTHONPATH:"./"
```

Example

Training a **FixMatch** with 32 filters on **cifar10** shuffled with seed=3, 10 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python fixmatch.py --filters=32 --dataset=cifar10.3@10-1 --train_dir ./experiments/fixmatch_pure
```

Training a **FixMatch + Reg** with 32 filters and 0.4 propagation regularizer weight on **cifar10** shuffled with seed=3, 10 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python fixmatch_reg.py --filters=32 --wpr=0.4 --dataset=cifar10.3@10-1 --train_dir ./experiments/fixmatch_reg
```

Training a **UDA** with 32 filters on **cifar10** shuffled with seed=3, 10 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python uda.py --filters=32 --dataset=cifar10.3@10-1 --train_dir ./experiments/uda_pure
```

Training a **UDA + Reg** with 32 filters and 0.4 propagation regularizer weight on **cifar10** shuffled with seed=3, 10 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python uda_reg.py --filters=32 --wpr=0.4 --dataset=cifar10.3@10-1 --train_dir ./experiments/uda_reg
```

Training a **FixMatch** with 32 filters on **cifar100** shuffled with seed=3, 100 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python fixmatch.py --filters=32 --dataset=cifar100.3@10-1 --train_dir ./experiments/fixmatch_pure
```

Training a **FixMatch + Reg** with 32 filters and 0.4 propagation regularizer weight on **cifar100** shuffled with seed=3, 100 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python fixmatch_reg.py --filters=32 --wpr=0.4 --dataset=cifar100.3@10-1 --train_dir ./experiments/fixmatch_reg
```

Training a **UDA** with 32 filters on **cifar100** shuffled with seed=3, 100 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python uda.py --filters=32 --dataset=cifar100.3@10-1 --train_dir ./experiments/uda_pure
```

Training a **UDA + Reg** with 32 filters and 0.4 propagation regularizer weight on **cifar100** shuffled with seed=3, 100 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python uda_reg.py --filters=32 --wpr=0.4 --dataset=cifar100.3@10-1 --train_dir ./experiments/uda_reg
```

Training a **FixMatch** with 32 filters on **SVHN** shuffled with seed=3, 10 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python fixmatch.py --filters=32 --dataset=svhn.3@10-1 --train_dir ./experiments/fixmatch_pure
```

Training a **FixMatch + Reg** with 32 filters and 0.4 propagation regularizer weight on **SVHN** shuffled with seed=3, 10 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python fixmatch_reg.py --filters=32 --wpr=0.4 --dataset=svhn.3@10-1 --train_dir ./experiments/fixmatch_reg
```

Training a **UDA** with 32 filters on **SVHN** shuffled with seed=3, 10 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python uda.py --filters=32 --dataset=svhn.3@10-1 --train_dir ./experiments/uda_pure
```

Training a **UDA + Reg** with 32 filters and 0.4 propagation regularizer weight on **SVHN** shuffled with seed=3, 10 labeled samples and 1 validation sample:

```
CUDA_VISIBLE_DEVICES=0 python uda_reg.py --filters=32 --wpr=0.4 --dataset=svhn.3@10-1 --train_dir ./experiments/uda_reg
```

Acknowledgement

Codebase from FixMatch.