Semantic Segmentation Tutorial using PyTorch

Semantic Segmentation Tutorial using PyTorch. Based on 2020 ECCV VIPriors Challange Start Code, implements semantic segmentation codebase and add some tricks.

Editer: Hoseong Lee (hoya012)

0. Experimental Setup

0-1. Prepare Library

```
pip install -r requirements.txt
```

0-2. Download dataset (MiniCity from CityScapes)

We will use MiniCity Dataset from Cityscapes. This dataset is used for 2020 ECCV VIPriors Challenge.

- workshop page: https://vipriors.github.io/challenges/
- challenge link: https://competitions.codalab.org/competitions/23712
- dataset download(google drive)
  - move dataset into `minicity` folder.

0-3. Dataset Simple EDA (Exploratory Data Analysis) - Class Distribution, Sample Distribution

benchmark class

```
    CityscapesClass('road', 7, 0, 'flat', 1, False, False, (128, 64, 128)),
    CityscapesClass('sidewalk', 8, 1, 'flat', 1, False, False, (244, 35, 232)),
    CityscapesClass('building', 11, 2, 'construction', 2, False, False, (70, 70, 70)),
    CityscapesClass('wall', 12, 3, 'construction', 2, False, False, (102, 102, 156)),
    CityscapesClass('fence', 13, 4, 'construction', 2, False, False, (190, 153, 153)),
    CityscapesClass('pole', 17, 5, 'object', 3, False, False, (153, 153, 153)),
    CityscapesClass('traffic light', 19, 6, 'object', 3, False, False, (250, 170, 30)),
    CityscapesClass('traffic sign', 20, 7, 'object', 3, False, False, (220, 220, 0)),
    CityscapesClass('vegetation', 21, 8, 'nature', 4, False, False, (107, 142, 35)),
    CityscapesClass('terrain', 22, 9, 'nature', 4, False, False, (152, 251, 152)),
    CityscapesClass('sky', 23, 10, 'sky', 5, False, False, (70, 130, 180)),
    CityscapesClass('person', 24, 11, 'human', 6, True, False, (220, 20, 60)),
    CityscapesClass('rider', 25, 12, 'human', 6, True, False, (255, 0, 0)),
    CityscapesClass('car', 26, 13, 'vehicle', 7, True, False, (0, 0, 142)),
    CityscapesClass('truck', 27, 14, 'vehicle', 7, True, False, (0, 0, 70)),
    CityscapesClass('bus', 28, 15, 'vehicle', 7, True, False, (0, 60, 100)),
    CityscapesClass('train', 31, 16, 'vehicle', 7, True, False, (0, 80, 100)),
```

```
        CityscapesClass('motorcycle', 32, 17, 'vehicle', 7, True, False, (0, 0, 230)),
        CityscapesClass('bicycle', 33, 18, 'vehicle', 7, True, False, (119, 11, 32)),
```

from 0 to 18 class, count labeled pixels

deeplab v3 baseline test set result

- Dataset has severe Class-Imbalance problem.
  - IoU of minor class is very low. (wall, fence, bus, train)

```
classes          IoU      nIoU
--------------------------------
road           : 0.963      nan
sidewalk       : 0.762      nan
building       : 0.856      nan
wall           : 0.120      nan
fence          : 0.334      nan
pole           : 0.488      nan
traffic light  : 0.563      nan
traffic sign   : 0.631      nan
vegetation     : 0.884      nan
terrain        : 0.538      nan
sky            : 0.901      nan
person         : 0.732    0.529
rider          : 0.374    0.296
car            : 0.897    0.822
truck          : 0.444    0.218
bus            : 0.244    0.116
train          : 0.033    0.006
motorcycle     : 0.492    0.240
bicycle        : 0.638    0.439
--------------------------------
Score Average  : 0.573    0.333
--------------------------------
```

1. Training Baseline Model

- I use DeepLabV3 from torchvision.
  - ResNet-50 Backbone, ResNet-101 Backbone
- I use 4 RTX 2080 Ti GPUs. (11GB x 4)
- If you have just 1 GPU or small GPU Memory, please use smaller batch size (<= 8)

```
python baseline.py --save_path baseline_run_deeplabv3_resnet50 --crop_size 576 1152 --batch_si
```

```
python baseline.py --save_path baseline_run_deeplabv3_resnet101 --model DeepLabv3_resnet101 --
```

1-1. Loss Functions

- I tried 3 loss functions.
  - Cross-Entropy Loss
  - Class-Weighted Cross Entropy Loss
  - Focal Loss
- You can choose loss function using `--loss` argument.
  - I recommend default (ce) or Class-Weighted CE loss. Focal loss didn'y work well in my codebase.

```
# Cross Entropy Loss
python baseline.py --save_path baseline_run_deeplabv3_resnet50 --crop_size 576 1152 --batch_si
```

```
# Weighted Cross Entropy Loss
python baseline.py --save_path baseline_run_deeplabv3_resnet50_wce --crop_size 576 1152 --batc
```

```
# Focal Loss
python baseline.py --save_path baseline_run_deeplabv3_resnet50_focal --crop_size 576 1152 --ba
```

1-2. Normalization Layer

- I tried 4 normalization layer.
  - Batch Normalization (BN)
  - Instance Normalization (IN)
  - Group Normalization (GN)
  - Evolving Normalization (EvoNorm)
- You can choose normalization layer using `--norm` argument.
  - I recommend BN.

```
# Batch Normalization
python baseline.py --save_path baseline_run_deeplabv3_resnet50 --crop_size 576 1152 --batch_si
```

```
# Instance Normalization
python baseline.py --save_path baseline_run_deeplabv3_resnet50_instancenorm --crop_size 576 11
```

```
# Group Normalization
python baseline.py --save_path baseline_run_deeplabv3_resnet50_groupnorm --crop_size 576 1152
```

```
# Evolving Normalization
python baseline.py --save_path baseline_run_deeplabv3_resnet50_evonorm --crop_size 576 1152 --
```

1-3. Additional Augmentation Tricks

- Propose 2 data augmentation techniques (CutMix, copyblob)
- CutMix Augmentation
  - Based on Original CutMix, bring idea to Semantic Segmentation.
- CopyBlob Augmentation
  - To tackle Class-Imbalance, use CopyBlob augmentation with visual inductive prior.
    - Wall must be located on the sidewalk
    - Fence must be located on the sidewalk
    - Bus must be located on the Road
    - Train must be located on the Road

```
# CutMix Augmentation
python baseline.py --save_path baseline_run_deeplabv3_resnet50_cutmix --crop_size 576 1152 --b
```

```
# CopyBlob Augmentation
python baseline.py --save_path baseline_run_deeplabv3_resnet50_copyblob --crop_size 576 1152 -
```

2. Inference

- After training, we can evaluate using trained models.
  - I recommend same value for `train_size` and `test_size`.

```
python baseline.py --save_path baseline_run_deeplabv3_resnet50 --batch_size 4 --predict;
```

2-1. Multi-Scale Infernece (Test Time Augmentation)

- I use [0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.2] scales for Multi-Scale Inference. Additionaly, use H-Flip.
  - Must use single batch (batch_size=1)

```
# Multi-Scale Inference
python baseline.py --save_path baseline_run_deeplabv3_resnet50 --batch_size 1 --predict --mst;
```

2-2. Calculate Metric using Validation Set

- We can calculate metric and save results into `results.txt` .
  - ex) My final validation set result

```
python evaluate.py --results baseline_run_deeplabv3_resnet50/results_val --batch_size 1 --pred
```

3. Final Result

-
- My final single model result is **0.6069831962012341**
  - Achieve 5th place on the leaderboard.
  - But, didn't submit short-paper, so my score is not official score.
- If i use bigger model and bigger backbone, performance will be improved.. maybe..
- If i use ensemble various models, performance will be improved!
- Leader board can be found in Codalab Challenge Page

4. Reference

- vipriors-challange-toolkit
- torchvision deeplab v3 model
- Focal Loss
- Class Weighted CE Loss
- EvoNorm
- CutMix Augmentation