A PyTorch Semantic Segmentation Toolbox

Zilong Huang[1,2], Yunchao Wei[2], Xinggang Wang[1], Wenyu Liu[1]
[1]School of EIC, HUST [2]Beckman Institute, UIUC

Abstract

In this work, we provide an introduction of PyTorch implementations for the current popular semantic segmentation networks, i.e. *DeeplabV3 [2] and PSPNet [9], which have achieved the competitive performance on various benchmarks. Our implementations provide a more simple, efficient and effective way to reproduce the reported results. By adopting Resnet101 as the backbone, our implementations of DeeplabV3 and PSPNet achieve 78.9% and 78.3% mIoU scores on the Cityscapes [3] under the single-scale testing scheme. The code is easily to be readable and modifiable. We hope our implementations can serve the researchers to make further improvements on the challenging semantic segmentation tasks. Source code is available at* `https://github.com/speedinghzl/pytorch-segmentation-toolbox`.

1. Introduction

Image semantic segmentation has always been one of fundamental research topics in computer vision. With the development of deep learning techniques, many approaches have been proposed to constantly boost the semantic segmentation results to new records. Most recently, two powerful methods, *i.e*. DeeplabV3 [2] and PSPNet [9], have achieved competitive segmentation results on many popular benchmarks, such as Cityscapes [3] and ADE20K [10]. However, the original implementations of these two works are based on Caffe [6], which is currently a bit difficult to install and get started compared with its following deep learning tools such as PyTorch and Tensorflow. More importantly, some results are hard to be reproduced unless taking the exact training settings, which are usually unaffordable for many researchers, *e.g*. the original PSPNet was trained on 16 P40 GPUs

To tackle the above mentioned issues as well as make the latest semantic segmentation techniques benefit more poverty researchers, we re-implement both DeeplabV3 and PSPNet using PyTorch, which is an open source machine learning library for Python and is becoming one of the most popular deep learning tools in the computer vision community. Our implementations are with the following advantages:

Table 1. Comparisons on w/ and w/o syn BN.

Method	w/o syn BN	w/ syn BN
PSPNet(ours)	76.10	78.30

- Integrating synchronous Batch Normalization (BN).
- Taking less time to train a converged model.
- Achieving better performance compared with the reported ones.

All these advantages enable our implementations to reproduce the semantic segmentation performance on many benchmarks with a much simpler and efficient way. More details are given in the following.

2. Highlights of Our Implementations

In this section, we provide the detailed advantages of our PyTorch implementations accompanied with several experimental results on Cityscapes using Resnet101 [4] as the backbone.

2.1. Synchronous BN

Synchronous BN [5] has been validated to be an effective trick, which can make a further improvement for various vision tasks. Rota et al. [1] proposed inplace-abn, which has implemented synchronous BN. We incorporate such an outstanding technique into our implementations. Specifically, synchronous BN can be easily utilized according to the following code:

```
import functools
from modules import InPlaceABN, InPlaceABNSync
BatchNorm2d = functools.partial(
                InPlaceABNSync,
                activation='none')

bn = BatchNorm2d(256) #the channel of features
```

As shown in Table 1, synchronous BN can provide 2.2% gain based on the PSPNet.

Table 2. Comparisons on training time.

Method	epoch	input size	batch size	total time
PSPNet (Original)	400	713	16 (16 P40)	$\sim 22h$
PSPNet (Ours)	108	768	8 (4 Titan V)	$\sim 17h$

Table 3. Comparison with the original results.

Method	PSPNet	DeeplabV3
Original	77.59	77.82
Ours	78.30	78.90

2.2. Fewness of Training Time

Our implementations can save a lot of time and computational resources to train a reliable semantic segmentation model. Comparisons upon the PSPNet are shown in Table 2. In general, our implementation cost less training time (17h *vs.* 22h) and GPUs (4 Titan V *vs.* 16 P40) compared with the original one. Thus, our implementations will significantly benefits the researchers who are not affordable to buy so many expensive GPUs.

2.3. Better Reproduced Performance

Beyond above mentioned excellent characteristics, our implementations can also achieve better reproduced performance. As shown in Table 3, our implementations outperform original ones by 1.71% (PSPNet) and 1.08% (DeeplabV3) on Cityscapes, respectively. We would like to provide comparisons on more popular benchmarks in the following updated version.

3. Benefits

Some recent projects have already benefited from our implementations. For example, Object Context Network (OCNet) [8] currently achieves the state-of-the-art results on Cityscapes and ADE20K. In addition, our code also make great contributions to Context Embedding with Edge Perceiving (CE2P) [7], which won the 1st places in all human parsing tracks in the 2nd LIP Challenge.

4. Acknowledgement

The code is heavily borrowed from pytorch-deeplab-resnet[1]. We greatly appreciate the efforts made by Isht Dwivedi.

References

[1] S. R. Bulò, L. Porzi, and P. Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. *CoRR, abs/1712.02616, December*, 5, 2017.

[2] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678, 2014.

[7] T. Ruan, T. Liu, Z. Huang, Y. Wei, S. Wei, Y. Zhao, and T. Huang. Devil in the details: Towards accurate single and multiple human parsing. In *AAAI*, 2019.

[8] J. W. Yuhui Yuan. Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*, 2018.

[9] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017.

[10] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.

[1] https://github.com/isht7/pytorch-deeplab-resnet