A PyTorch Semantic Segmentation Toolbox

Zilong Huang\textsuperscript{1,2}, Yunchao Wei\textsuperscript{2}, Xinggang Wang\textsuperscript{1}, Wenyu Liu\textsuperscript{1}

\textsuperscript{1}School of EIC, HUST \textsuperscript{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 \cite{2} and PSPNet \cite{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 \cite{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 \url{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 \cite{2} and PSPNet \cite{9}, have achieved competitive segmentation results on many popular benchmarks, such as Cityscapes \cite{3} and ADE20K \cite{10}. However, the original implementations of these two works are based on Caffe \cite{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:

- 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 \cite{4} as the backbone.

2.1. Synchronous BN

Synchronous BN \cite{5} has been validated to be an effective trick, which can make a further improvement for various vision tasks. Rota et al. \cite{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:

```python
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>
<thead>
<tr>
<th>Method</th>
<th>w/o syn BN</th>
<th>w/ syn BN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPNet(ours)</td>
<td>76.10</td>
<td>78.30</td>
</tr>
</tbody>
</table>

Table 1. Comparisons on w/ and w/o syn BN.
Table 2. Comparisons on training time.

<table>
<thead>
<tr>
<th>Method</th>
<th>epoch</th>
<th>input size</th>
<th>batch size</th>
<th>total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPNet (Original)</td>
<td>400</td>
<td>713</td>
<td>16 (16 P40)</td>
<td>∼22h</td>
</tr>
<tr>
<td>PSPNet (Ours)</td>
<td>108</td>
<td>768</td>
<td>8 (4 Titan V)</td>
<td>∼17h</td>
</tr>
</tbody>
</table>

Table 3. Comparison with the original results.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSPNet</th>
<th>DeeplabV3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>77.59</td>
<td>77.82</td>
</tr>
<tr>
<td>Ours</td>
<td>78.30</td>
<td>78.90</td>
</tr>
</tbody>
</table>

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 benefit 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-resnet1. We greatly appreciate the efforts made by Isht Dwivedi.

References


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