Project 6 (Extra Credit): Semantic Segmentation Deep Learning

CS 6476

Spring 2021

Brief

- Due: May 5, 2021 11:59PM
- Project materials including report template: proj6.zip
- Hand-in: through Gradescope
- Required files: `<your_gt_username>.zip`, `<your_gt_username>_proj6.pdf`

Overview

In this project, you will design and train deep convolutional networks for semantic segmentation.

Setup

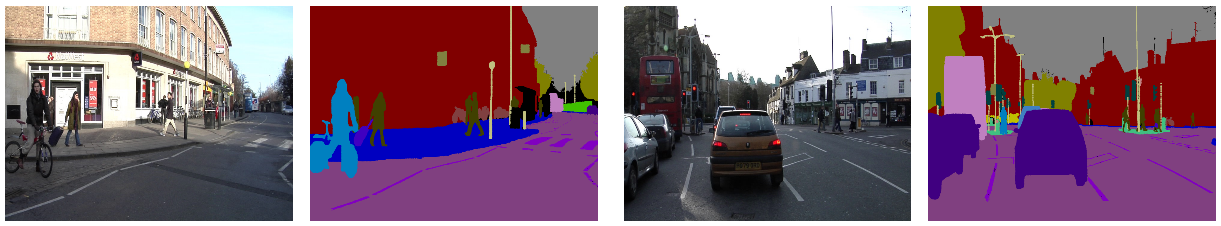
1. Install Miniconda. It doesn't matter whether you use Python 2 or 3 because we will create our own environment that uses python3 anyways.
2. Download and extract the project starter code.
3. Create a conda environment using the appropriate command. On Windows, open the installed "Conda prompt" to run the command. On MacOS and Linux, you can just use a terminal window to run the command, Modify the command based on your OS (`linux`, `mac`, or `win`): `conda env create -f proj6_env_<OS>.yml`
4. This will create an environment named "cs6476_proj6". Activate it using the Windows command, `activate cs6476_proj6` or the MacOS / Linux command, `conda activate cs6476_proj6` or `source activate cs6476_proj6`
5. Install the project package, by running `pip install -e .` inside the repo folder. This might be unnecessary for every project, but is good practice when setting up a new `conda` environment that may have `pip` requirements.
6. Run the notebook using `jupyter notebook ./proj6_code/proj6.ipynb`
7. After implementing all functions, ensure that all sanity checks are passing by running `pytest proj6_unit_tests` inside the repo folder.
8. Generate the zip folder for the code portion of your submission once you've finished the project using `python zip_submission.py --gt_username <your_gt_username>`

Dataset

The dataset to be used in this assignment is the Camvid dataset, a small dataset of 701 images for self-driving perception. It was first introduced in 2008 by researchers at the University of Cambridge [1]. You can read more about it at the original dataset page or in the paper describing it. The images have a typical size of around 720 by 960 pixels. We'll downsample them for training though since even at 240 x 320 px, most of the scene detail is still recognizable.

Today there are much larger semantic segmentation datasets for self-driving, like Cityscapes, WildDashV2, Audi A2D2, but they are too large to work with for a homework assignment.

The original Camvid dataset has 32 ground truth semantic categories, but most evaluate on just an 11-class subset, so we'll do the same. These 11 classes are 'Building', 'Tree', 'Sky', 'Car', 'SignSymbol', 'Road', 'Pedestrian', 'Fence', 'Column_Pole', Sidewalk', 'Bicyclist'. A sample collection of the Camvid images can be found below:



(a) Image A, RGB (b) Image A, Ground Truth (c) Image B, RGB (d) Image B, Ground Truth

Figure 1: Example scenes from the Camvid dataset. The RGB image is shown on the left, and the corresponding ground truth "label map" is shown on the right.

1 Implementation

For this project, the majority of the details will be provided into two separate Jupyter notebooks. The first, `proj6_local.ipynb` includes unit tests to help guide you with local implementation. After finishing that, upload `proj6_colab.ipynb` to Colab. Next, zip up the files for Colab with our script `zip_for_colab.py`, and upload these to your Colab environment.

We will be implementing the PSPNet [3] architecture. You can read the original paper here. This network uses a ResNet [2] backbone, but uses *dilation* to increase the receptive field, and aggregates context over different portions of the image with a "Pyramid Pooling Module" (PPM).

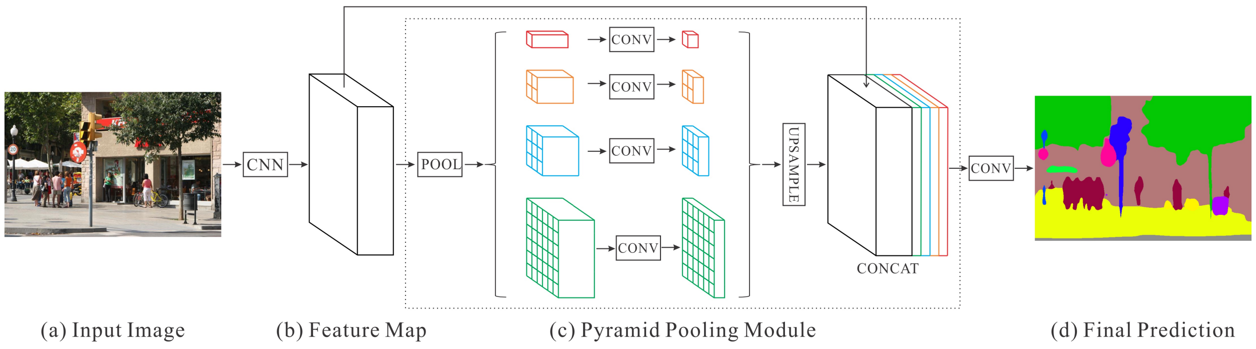
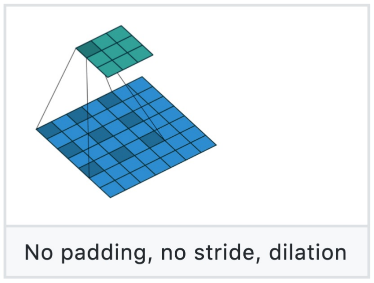
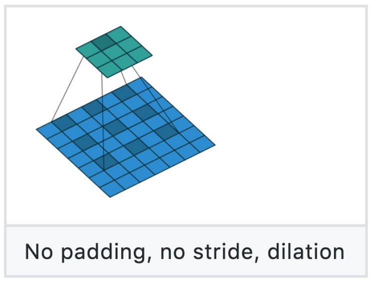


Figure 2: PSPNet architecture. The Pyramid Pooling Module (PPM) splits the $H \times W$ feature map into KxK grids. Here, 1×1, 2×2, 3×3, and 6×6 grids are formed, and features are average-pooled within each grid cell. Afterwards, the 1×1, 2×2, 3×3, and 6×6 grids are upsampled back to the original $H \times W$ feature map resolution, and are stacked together along the channel dimension.

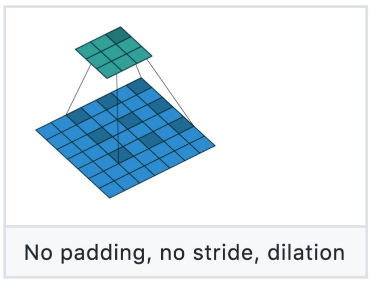
You can read more about dilated convolution in the Dilated Residual Network here, which PSPNet takes some ideas from. Also, you can watch a helpful animation about dilated convolution here.



(a)



(b)



(c)

Figure 3: Dilation convolution. Figure source: https://github.com/vdumoulin/conv_arithmetic#dilated-convolution-animations

Suggested order of experimentation

1. Start with just ResNet-50, without any dilation or PPM, end with a 7×7 feature map, and add a 1×1 convolution as a classifier. Report the mean intersection over union (mIoU).
2. Now, add in data augmentation. Report the mIoU. (you should get around 48% mIoU in 50 epochs, or 56% mIoU in 100 epochs, or 58-60% in 200 epochs).
3. Now add in dilation. Report the mIoU.
4. Now add in PPM module. Report the mIoU.
5. Try adding in auxiliary loss. Report the mIoU (you should get around 65% mIoU over 100 epochs, or 67% in 200 epochs).

Rubric

- +4 pts: Part 1 Code
- +4 pts: Part 2 Code

- +4 pts: Part 3 Code
- +4 pts: Part 4 Code
- +20 pts: Part 5 Code (you'll need to reach 64% mIoU on your submitted grayscale PNG label maps to get full credit)
- +14 pts: Report
- -5*n pts: Lose 5 points for every time you do not follow the instructions for the hand-in format

Submission format

This is very important as you will lose 5 points for every time you do not follow the instructions. You will submit two items to Gradescope:

1. `<your_gt_username>.zip` containing:
   (a) `proj6_code/` - directory containing all your code for this assignment
   (b) `grayscale_predictions.zip` - a zip file with your model's outputs on the validation set (as PNG files). See the Colab notebook for more details.
2. `<your_gt_username>_proj6.pdf` - your report

Do ***not*** install any additional packages inside the conda environment. The TAs will use the same environment as defined in the config files we provide you, so anything that's not in there by default will probably cause your code to break during grading. Do ***not*** use absolute paths in your code or your code will break. Use relative paths like the starter code already does. Failure to follow any of these instructions will lead to point deductions. Create the zip file using `python zip_submission.py --gt_username <your_gt_username>` (it will zip up the appropriate directories/files for you!) and hand it in with your report PDF through Gradescope (please remember to mark which parts of your report correspond to each part of the rubric).

Credits

Assignment developed by John Lambert and James Hays.

References

[1] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. "Semantic Object Classes in Video: A High-definition Ground Truth Database". In: *Pattern Recogn. Lett.* 30.2 (2009).

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* June 2016.

[3] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. "Pyramid Scene Parsing Network". In: *CVPR.* 2017.