Project 5: Semantic Segmentation Deep Learning

CS 4476

Spring 2024

Brief

- Due: Check Canvas for up to date information
- Project materials including report template: Github
- Hand-in: through Gradescope
- Required files: `<your_gt_username>.zip`, `<your_gt_username>_proj5.pdf`

Overview

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

Setup

1. Follow Project 5 Github README.
2. Run the notebook using `jupyter notebook proj5_local.ipynb` inside the repository folder.
3. After implementing all functions, ensure that all sanity checks are passing by running `pytest tests` inside the repository folder.
4. After passing the unit tests, run `python zip_for_colab.py` which will create `cv_proj5_colab.zip`
5. Upload `proj5_colab.ipynb` to Colab.
6. Upload `cv_proj5_colab.zip` to session storage.
7. Click `Run All` inside `Runtime` which will train your model.

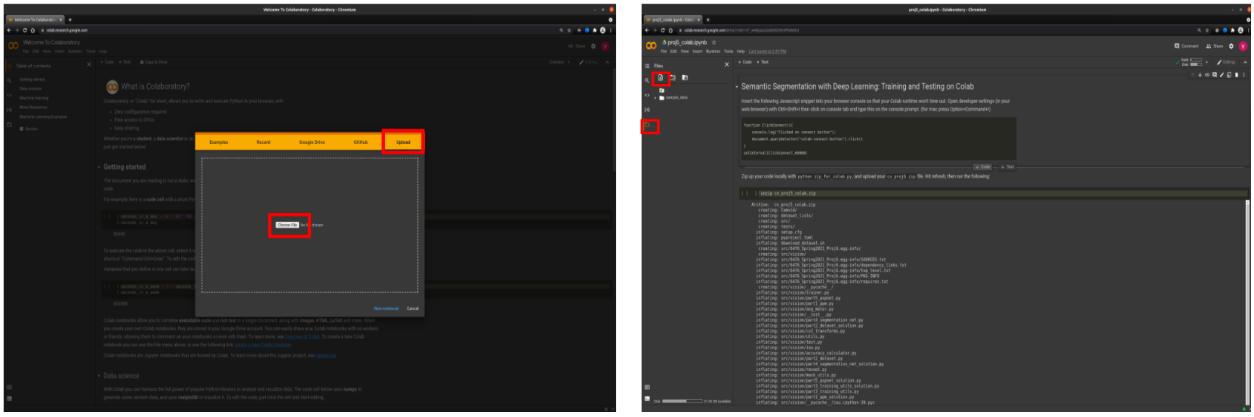


Figure 1: Figure describing setup steps 5-7.

8. Record the results and download `grayscale_predictions.zip`.

9. 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:

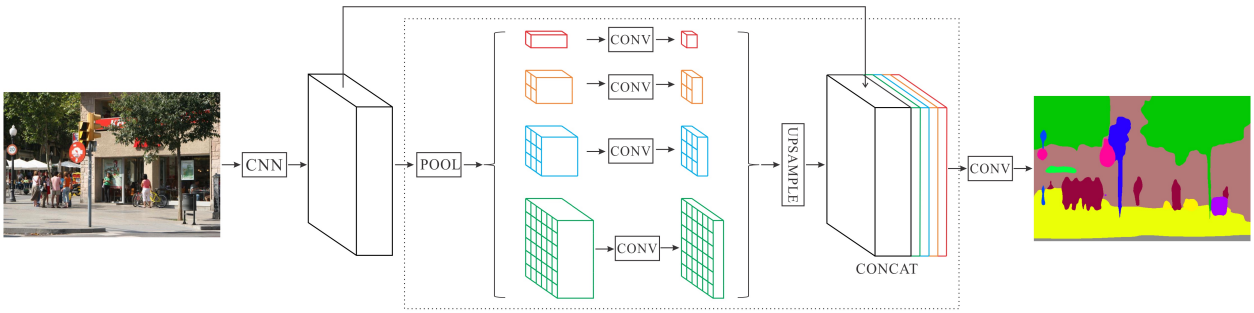
(a) Image A, RGB (b) Image A, Ground Truth (c) Image B, RGB (d) Image B, Ground Truth

Figure 2: 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, `proj5_local.ipynb` includes unit tests to help guide you with local implementation. After finishing that, upload `proj5_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.

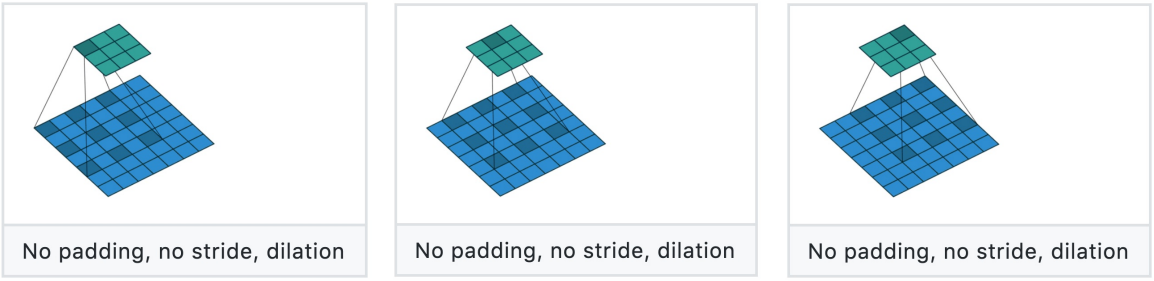
We will be implementing the PSPNet [3] architecture. You can read the original paper here. This network uses a ResNet [2] backbone, but uses *dilation* to increase the receptive field, and aggregates context over different portions of the image with a "Pyramid Pooling Module" (PPM).



(a) Input Image (b) Feature Map (c) Pyramid Pooling Module (d) Final Prediction

Figure 3: 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 4: 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).
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 63% mIoU over 50 epochs, or 65% in 100 epochs).

WandB (Extra Credit)

In this section, you will incorporate WandB in your training loop. Specifically, we want you to log the metrics from each epoch of training. Make the relevant changes in main_worker_wandb() in src/vision/trainer.py. There is no test case for this section, we just expect you to add a link to your WandB project in the report. Make sure you project is public.

Transfer Learning (Extra Credit)

Use the PSPNet trained on Camvid dataset as your pre-trained model and train it on KITTI road segmentation dataset. Finish the function `model_and_optimizer` in part 6. Train the pre-trained model on KITTI. Report the mIoU that shows how well the model was able to transfer to KITTI dataset (you must get 70% to receive full credit). You need to submit the `training_results_dict.json` file which is generated after performing transfer learning.

Writeup

For this project (and all other projects), you must do a project report using the template slides provided to you. Do ***not*** change the order of the slides or remove any slides, as this will affect the grading process on Gradescope and you will be deducted points. In the report you will describe your algorithm and any decisions you made to write your algorithm a particular way. Then you will show and discuss the results of your algorithm. The template slides provide guidance for what you should include in your report. A good writeup doesn't just show results–it tries to draw some conclusions from the experiments. You must convert the slide deck into a PDF for your submission, and then assign each PDF page to the relevant question number on Gradescope.

If you choose to do anything extra, add slides *after the slides given in the template deck* to describe your implementation, results, and analysis. You will not receive full credit for your extra credit implementations if they are not described adequately in your writeup.

Bells & whistles (extra credit)

Implementation of bells whistles can increase your grade on this project by up to 10 points (potentially over 100). The max score is 110.

- +7 pts: Transfer Learning
- +3 pts: WandB Integration

Rubric

CS 4476

- +10 pts: Part 1 Code
- +5 pts: Part 2 Code
- +10 pts: Part 3 Code
- +15 pts: Part 4 Code
- +40 pts: Part 5 Code (+32 pts: PSPNet model accuracy. you'll need to reach 60% mIoU on your submitted grayscale PNG label maps to get full credit. You will get partial credit if your mIoU is lower than the threshold. score = mIoU/threshold $\times$ 32)
- +10 pts: Extra Credits
- +20 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) `src/vision/` - directory containing all your code for this assignment
   (b) `setup.cfg`: setup file for environment, no need to change this file
   (c) `grayscale_predictions.zip`: after running Colab, this zip file will be created with your model's outputs on the Camvid validation set (remember to download before closing your session!)
   (d) `training_results_dict.json` : (extra credit) JSON file generated during Transfer Learning on KITTI dataset (remember to download before closing your session!)
   (e) `additional_data/` : (optional) if you use any data other than the images we provide you, please include them here
   (f) `README.txt`: (optional) if you implement any new functions other than the ones we define in the skeleton code (e.g., any extra credit implementations), please describe what you did and how we can run the code. We will not award any extra credit if we can't run your code and verify the results.
2. `<your_gt_usernamme>_proj5.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

This assignment was developed by Humphrey Shi, Prateek Garg, Manushree Vasu and Aditya Kane, based on similar projects by James Hays and Judy Hoffman.

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.