CSE 252C Assignment 3
Advanced Computer Vision
Spring 2019
Due June 16, 11pm PST

Instructions :

- Attempt all questions.
- Please comment all your code adequately.
- Turn-in a PDF for the written report and a zip-folder for the code. The report should be called `LastName_FirstName.pdf` and similarly the zip folder should be `LastName_FirstName.zip`.
- Include all relevant information such as text answers, output images and main code snippets in the PDF.
- Include the following code in the zip file.
  1. `model.py`, `train.py`, `dataLoader.py` and `test.py` for Section A.
- Submit your homework by Gradescope. Remember to correctly select pages for each answer on Gradescope to allow proper grading.

The homework consists of two parts. In the first part, you will train a UNet for semantic segmentation on PASCAL VOC 2012 dataset [1]. In the second part, you will test SSD [2] on PASCAL VOC 2012 dataset and analyze its differences from other popular object detection frameworks [3].

A UNet for Image Segmentation

In this homework, we will provide the dataset loader, the evaluation code, the basic UNet structure and some useful functions. You will be asked to try different variations of network structure and decide the best training strategies to obtain good results. Like in previous homeworks, you are welcome to cite any open source codes that help you improve performance.

1. **Download code.** The code is contained in the folder `Segmentation.zip`. The unzipped folder includes:
   (a) `test.py`: The file for evaluation.
   (b) `dataLoader.py`: The file to load the data for training and testing.
   (c) `model.py`: The file for models. The residual block (`ResBlock`) and the code to load pretrained weights of `resnet18` (`loadPretrainedWeight`) are given. The basic encoder and decoder are also given as a reference.

(d) `colormap.mat`: The color map used to visualize segmentation results.

(e) `utils.py`: The file for two useful functions. The `computeAccuracy` function computes the unnormalized confusion matrix of each batch of labels. The `save_label` function turns the label into an image using the given color map and saves the image at the assigned location. Also see `test.py` for how these two functions are being used.

(f) `train.py`: An empty file where you will implement your training script.

2. **Implement the network structure.** You are required to implement 2 versions of UNet structure since the basic structure has already been given. In all three versions, the `resnet18` structure before average pooling and fully connected layer will be used as the building block for encoder. You are strongly recommended to use weights pretrained on ImageNet, which may have a major impact on the performance.

   (a) **Basic UNet**: The code is given as a reference. Please see `encoder` and `decoder` class in `train.py`. The `encoder` comes from `resnet18` and the decoder consists of transpose convolutional layers and bilinear interpolation layers so that the final output will be of the same size as the image. Skip links are added to help the network recover more details. Please do not change the encoder. However, you are free to change the decoder, while ensuring that the structure of your decoder across three versions of the networks are similar so that you can make a fair comparison of their performances.

   (b) **UNet with dilation**: We modify the encoder to a dilated `resnet18` as described in Section 2 of [4][1]. We set the stride of the last 4 residual blocks to be 1 so that the highest level feature maps will be 4×4 times larger. To increase the receptive field, we set the dilation of residual blocks that are fourth and third from the end to be 2, while the dilation of the residual blocks that are first and second from the end are set to 4. The decoder should be modified accordingly. Implement your new encoder and decoder under class `encoderDilation` and `decoderDilation`. Ensure that for images of arbitrary shapes, the decoder will give segmentation outputs of the same shape. **[15 points]**

   (c) **UNet with dilation and pyramid pooling**: Based on the encoder-decoder structure with dilation, add pyramid pooling layer after the last residual block of encoder. Implement the pyramid pooling layer following [5]. Notice that after adding the pyramid layer, the number of channels of the output feature to the first transpose convolutional layer will change from 512 to 1024. Please implement your new encoder and decoder under classes `encoderSPP` and `decoderSPP`, respectively. **[15 points]**

3. **Implement training script and train the network.** Train your network using 1464 images from the training set of PASCAL VOC 2012. If you are not familiar with training scripts, you can refer to `test.py` in this homework and `casia_train.py` in the previous homework. The structures of the training script are very similar. Please remember to output the training loss and training accuracy which may help you find the best hyper parameters. **[40 points]**

[1]You are not required to consider degridding in Section 4 of [4].

The followings are suggestions to which you may need to pay attention:

(a) To accelerate the training speed, you can use the Pytorch multi-threaded data loader. **Important:** if you use multi-threaded data loader, remember to either randomly shuffle the data or change the random seeds after every epoch. Otherwise you will have severe overfitting issues because the data loader will always crop the same region of the image.

(b) It is recommended to compute the prediction mIoU every epoch, since the curve of mIoU can be very different from the inverse of loss function. It may help you find the best training strategy.

(c) To overcome over-fitting issues, you are encouraged to adopt more aggressive data augmentation methods, such as flipping the images or changing the intensity.

(d) There are many things that may influence performance, such as learning rate, batch size and network structure of encoder and decoder. It might be hard to achieve state-of-the-art results. **The grading of the homework will not focus on the final mean IoU but more on analysis.** So don't be too worried if you cannot get very good performance. Just make sure that you describe what you observe and answer the questions succinctly.

4. **Answer the following questions:**

   (a) Describe the loss function you use to train the semantic segmentation network. If you change the structure of the decoder, describe your new network architecture. **[10 points]**

   (b) Describe your training details, such as: what kind of optimizer is used to train the network, what's the learning rate and the batch size, whether you decrease the learning rate as the training progresses, number of epochs required to train the network, or any other details that you find important for performance. Note that in order to compare the three network structures, learning details for them should be the same. **[10 points]**

   (c) Draw the loss curves of the three network structures in the same graph. **[10 points]**

   (d) Evaluate the trained models using `test.py`.

      i. **Basic UNet:** `python test.py`. The testing mean IoU of 21 categories of object are stored in `test/accuracy.npy`. You can add flags if necessary.

      ii. **UNet with dilation:** `python test.py −−isDilation`. The testing mean IoU of 21 categories of objects are stored in `test_dilation/accuracy.npy`. You can add flags if necessary.

      iii. **UNet with dilation and pyramid pooling:** `python test.py −−isSpp`. The testing mean IoU of 21 categories of object are stored in `test_spp/accuracy.npy`. You can add flags if necessary.

   Draw a table to summarize quantitative performances of the 3 variations of the UNet structure. The table should include the mean IoU of 21 categories of objects and their average mean IoU. **[10 points]**

(e) Make a figure for qualitative comparisons of the 3 methods, shown on 4 different input images. Please show the segmentation results for the same image but different networks so the differences can be compared. Briefly describe the results you obtain and any observations. **[10 points]**

(f) Explain your observations in terms of: (i) what choices helped improve the accuracy and (ii) what other steps could have been tried to further improve accuracy? **[10 points]**

B SSD Object Detection

In this section, you will try the trained model of SSD [2] on the PASCAL VOC 2012 dataset.

1. Download `Detection.zip` from the course website, which is modified from `https://github.com/amdegroot/ssd.pytorch`. Run `eval.py` code to get the object detection average precision (AP) on the PASCAL VOC 2012 dataset. The model is already trained on the PASCAL VOC 2012 object detection dataset. Draw a table in your report summarizing the AP of all 20 object categories and their mean. **[10 points]**

2. Answer the following questions:

   (a) Briefly explain how average precision is computed for PASCAL VOC 2012 dataset. Please check the code (`eval.py : Line 163 − 191`) since there are different ways to compute average precision. **[10 points]**

   (b) Explain how SSD can be much faster compared to Faster RCNN [3] ? **[10 points]**

   (c) Usually the number of negative bounding boxes (boxes without any object) is much larger than the number of positive bounding boxes. Explain how this imbalance is handled in SSD and Faster RCNN, respectively. **[10 points]**

3. Randomly pick up some images from the PASCAL VOC 2012 dataset and some from other sources. Visualize the bounding box prediction results and include a figure in your report. You can use the code in folder `demo` for visualization. **[10 points]**

References

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[4] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480, 2017.

[5] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.