HOMEWORK ASSIGNMENT # 6

DUE: Tuesday, November 27, 2018, 5PM

CSCI677: Computer Vision, Prof. Nevatia

Fall Semester, 2018

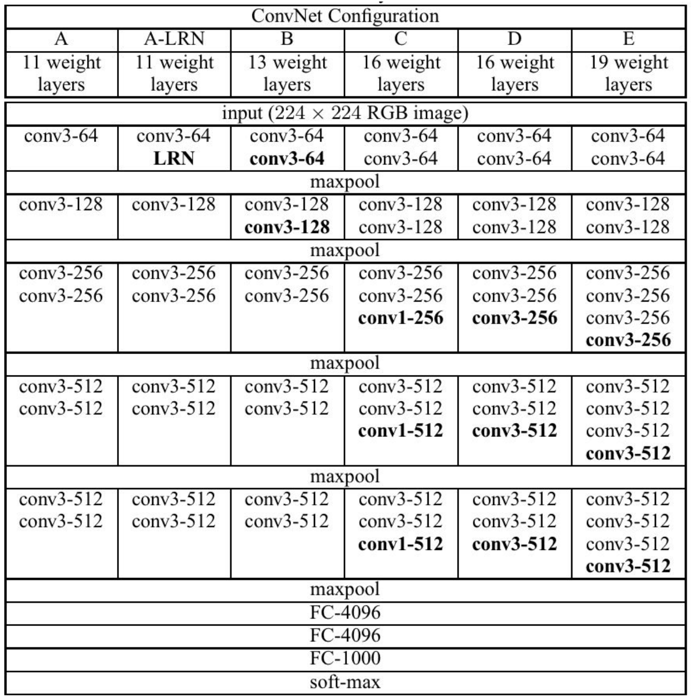
This is a programming assignment to create, train, and test a CNN for the task of semantic segmentation following the FCN32s method described in: http://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf .

a) **Dataset**
The dataset we use for this assignment is based on KITTI Road semantic segmentation dataset. There are three folders in the **dataset we provide**. There are 289 images of road scenes in the **image** folder and 289 semantically annotated images in the **gt** folder. Each image is a 352x1216 RGB image. In the folder named **label**, there are 289 binary files, each file stores a (352,1216) numpy array representing the per-pixel numerical label of a image. In this dataset, **0** for **Non-Road**, **1** for **Road**, and **-1** for **Void.**
You can relate the road scene image and its corresponding annotated image/label map by their filenames. For example, $PATH_TO_DATASET/image/test/um_000000.png is paired with $PATH_TO_DATASET/gt/test/um_road_000000.png and $PATH_TO_DATASET/label/um_road_000000.label.
You can use *cv2.imread($*FILENAME*)* to read the images and *pickle.load(open($*FILENAME*,"rb"))* to read the binary label files. You only need images in the **image** folder and labels in the **label** folder for training/validation/testing. Images in the **gt** folder helps you visually evaluate your results.
Please split the images in the **train** folder in the **image** folder into two sets, train set and val set. Train set should contain 199 images. Val set should contain 45 images. Use the 45 images in the **test** folder for testing.

b) **Architecture**:
Construct a Fully Convolutional Network FCN-32s. FCN-32s network is based on VGG-16 network with 3 modifications:
1. Replace the first fc layer by a convolutional layer with kernel size of 7. The number of the kernels is the output dimension of the original fc layer, which is **4096**;
2. Replace the second and third fc layers by convolutional layers with kernel size of 1. The numbers of kernels are **4096** and **1** respectively.
3. Add a deconvolutional layer after the third fc layer. In tensorflow, you may use `tf.layers.conv2d_transpose()` to implement the deconvolutional layer. You can find the usage of the function in: https://www.tensorflow.org/api_docs/python/tf/layers/conv2d_transpose. The configuration for the deconvolutional layer in this assignment is: stride=32, kernel

size = 64, filter =1.

Except for the three modifications above, other parts of FCN-32s are as same as VGG-16, below is the table from VGG paper(https://arxiv.org/pdf/1409.1556.pdf), please follow the **ConvNet Configuration D** showed in the table and apply the modifications mentioned above to build your FCN-32s.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Please note:

1. The kernel sizes of the maxpool processes in the table are all **2**.
2. All the convolutional processes should be followed by a ReLU process (including the three layers mentioned in the modification part).
3. All the padding mode should be set as "SAME". For example:

   tf.nn.max_pool(input_tensor, kernel_size, padding = 'SAME');

   tf.nn.conv2d(input_tensor, kernel_size, padding = 'SAME');

   It will help you to make sure the output size of conv layer will be the same as the input size and the output size if the maxpool layer will be exactly the 1/4 of the input size.

Details of FCN-32s can be found in the paper cited above.

c) **Evaluation metric**:

We are going to use pixel-level intersection-over-union (IoU) metrics to evaluate the network output. Pixel-level IoU = TP/(TP+FP+FN), where TP, FP, and FN are the numbers of true positive, false positive, and false negative pixels, respectively. Please write your own code to evaluate the network performance. Please calculate the IoU only for **Road** category.

c) **Training**:

Train the network using the images in the train set mentioned in the **Dataset** section. We suggest using a batch size of 1(one image already includes large number of pixel samples), SGD optimizer with momentum(tf.train.MomentumOptimizer), learning rate = 0.001, momentum = 0.99. though you are free to experiment with other values. We use tf.nn.sigmoid_cross_entropy_with_logits() as the loss function. Please make sure that when you calculate the loss you **ignore the pixels labeled as <u>Void</u> in the ground truth label maps**. One possible way is masking out the pixels whose gt labels are smaller than 0.

As in HW5, you may use TensorFlow routines to train the network. We recommend that you write your own sampling program for constructing mini-batches. No augmentation of training data is recommended.

Record the IoU and loss after each iteration so that you can monitor it and plot it to show results.

d) Results

Map your output label maps to color images which should look like the images in the **gt** folder. Please use the RGB code **[255,0,255]** for pixels labeled as **Road**, **[255,0,0]** for pixels labeled as **Non-Road**. (Please note that default color format in opencv is BGR)

SUBMISSION:

You should turn in a single PDF with the following:

1. A brief description of the programs you write, including the source listing.

2. Show evolution of loss function with multiple steps.

3. A summary and discussion of the results, including effects of parameter choices. Include visualization of results; show some examples of successful and some failure examples.