Deep Learning Lab Course

Gabriel Leivas Oliveira, Tonmoy Saikia

Due: December 4th, 2018

The goal of this exercise is to understand the main concepts and components of a decoder module of Fully Convolutional Networks (FCNs) for semantic segmentation. At the end send us a small report (1 or 2 pages). The report should include the following:

1. A plot of Intersection Over Union(IoU) vs epochs for each decoder configuration.
2. A table with maximum IoU value for each configuration.

In total there are four configurations to be implemented (described in the sections below). Please consult **Test_Net.py** to generate IoUs.

1 Implementing a decoder module for FCNs

First, we will implement a simple decoder module to upsample the final features obtained from the encoder. Our FCN encoder consists of four convolution layers with stride 2 which will produce a feature map whose spatial resolution is 16× smaller than the input image. The first task will be to implement a module to upsample the features back to the original spatial resolution with a single module. To achieve this, transposed convolution with stride similar to the upsampling rate must be used. This experiment will be called *Configuration 1*. In order to implement this and all other configurations (described below) we will use the file **nets_definition.py**.

2 Hierarchical refinement and impact of skip connections

After you have implemented a single stage decoder to upsample the encoder features back to image resolution you should look at multi-stage decoder for upsamling.

One important component of encoder decoder networks is a skip connection. Before we proceed to multi-stage decoders we will restructure *Configuration 1* to include a skip connection. Figure 1 shows the structure of a refinement block

to be implemented. It consists of the upsampling module you implemented previously, which is composed of a transposed convolution and an activation function (ELU). After that you must check if the upsampled features and skip connection features are of the same spatial resolution. If not crop the largest one and concatenate both before sending to a convolution layer to fuse the feature maps.

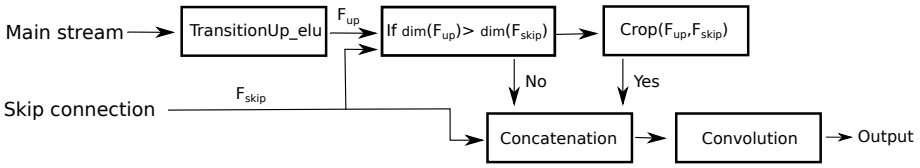

Figure 1: The refinement module to be implemented.

Once the refinement block is implemented, we can now construct a multi-stage decoder with skip connections. The effect of stacking multiple refinement blocks and different upsampling configurations should be studied. You should stack one, two and three refinement blocks as shown in Figure 2 (We will call these *Configuration 2,3,4* respectively). The last upsamling step in each configuration does not use a skip connection (It is the same as *Configuration 1*).

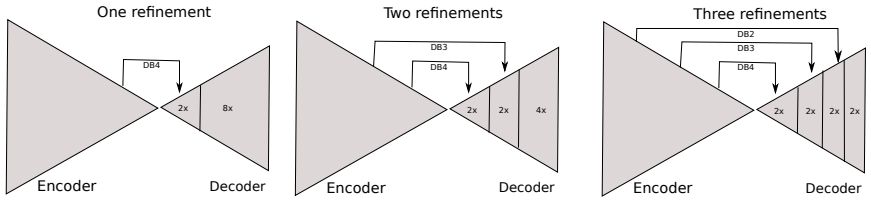

Figure 2: *Configurations 2, 3, 4* to be implemented. Each configuration include a new refinement module with upsampling rate of 2× and its corresponding skip-connection.

3 Dataset

The used dataset is the CamVid semantic segmentation dataset. The dataset is constituted by 468 training images and 233 testing images. The dataset is divided in 11 classes: Sky(0), Building(1), Pole(2), Road(3), Sidewalk(4), Tree(5), Sign(6), Fence(7), Car(8), Pedestrian(9), Cyclist(10).

Data must me downloaded following the readme file provide at the data folder.

4 Training and Testing

We must fill the gaps at **nets_definition.py** for the four different configurations. See **readme.me** for the proper use of the training and testing code, respectively **Train_Net.py** and **Test_Net.py**.