Fully Convolutional Networks

Fully Convolutional Networks (FCNs) owe their name to their architecture, which are built only from locally connected layers, such as convolution, pooling and upsampling. Note that no dense layer is used in this kind of architecture. This reduces the number of parameters and computation time. Also, the network can work regardless of the original image size, without requiring any fixed number of units at any stage, given that all connections are local. To obtain a segmentation map (output), segmentation networks usually have 2 parts:

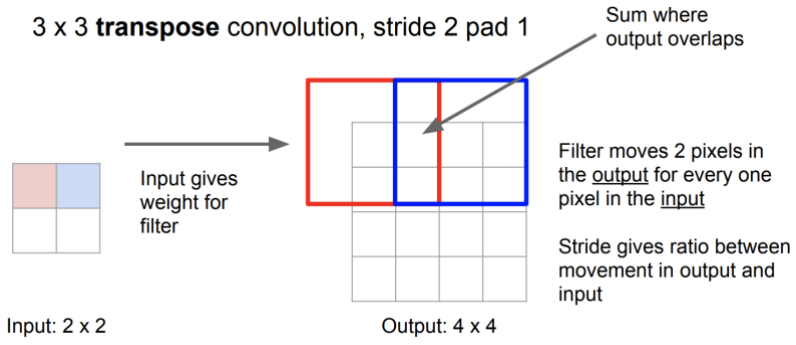
- Downsampling path : capture semantic/contextual information
- Upsampling path : recover spatial information

The **downsampling path** is used to extract and interpret the context (*what*), while the **upsampling path** is used to enable precise localization (*where*). Furthermore, to fully recover the fine-grained spatial information lost in the pooling or downsampling layers, we often use skip connections.

A skip connection is a connection that bypasses at least one layer. Here, it is often used to transfer local information by concatenating or summing feature maps from the downsampling path with feature maps from the upsampling path. Merging features from various resolution levels helps combining context information with spatial information.

An FCN transforms the height and width of the intermediate layer feature map back to the size of input image through the transposed convolution layer, so that predictions have a one-to-one correspondence with input images in spatial dimension *(width and height*). Given a position on the spatial dimension, the output of the channel dimension will be a category prediction of the pixel corresponding to the location. Common evaluation metrics for FCN are Pixel accuracy and mean loU.

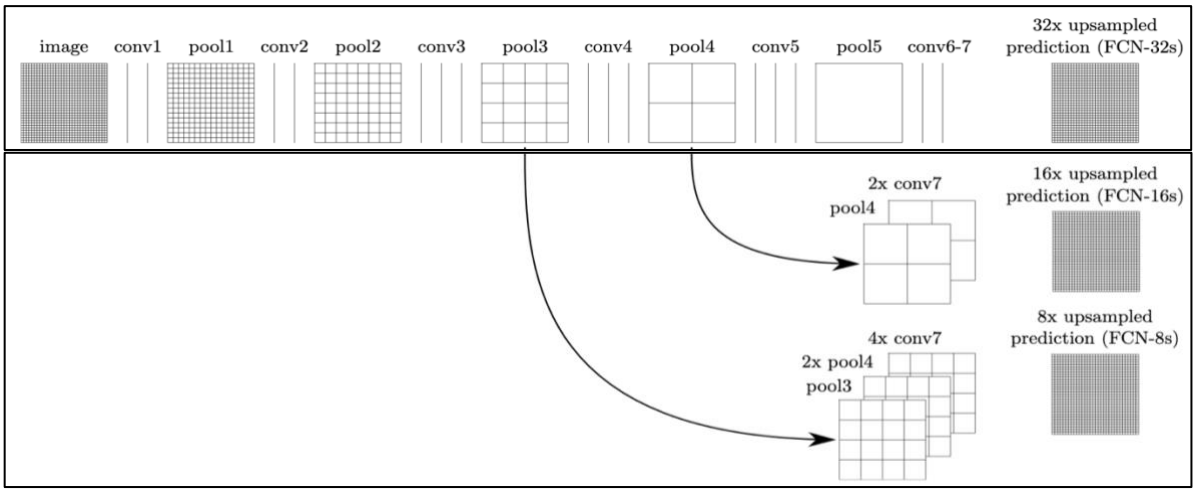
FCN consists of two folds. First, CNNs shrink features to shape of $"1 \times 1 \times Channel"$. Second, using transposed convolution to upsample outputs the segmentation mask. Finally, it applied softmax at the end for pixel classification. First fold is the downsampling, which use pooling and **strided convolution** to subsample features space to assign importance (learnable weights and biases) to various aspects /objects in the image and be able to differentiate one from the other. Second fold is the upsampling, which use unpooling or strided transpose convolution to upsample feature maps to match the original image size. It also utilizes **skip architecture** to refine predictions by combining features from earlier layers with outputs from **transposed convolution**. In terms of transposed convolution, output contains copies of filtered weighted by the input, summing at where at overlaps in output. Kernel weights are usually trained using gradient descent and initialized with bilinear interpolation. In below graph, it is a 3×3 transpose convolution with stride 2 and pad 1 (stride gives ratio between movement in input and output).



An example of transposed convolution (upsampling)

Next, I will use sematic segmentation as an example to illustrate how FCN works in more detail. Compared

with classification and detection tasks, segmentation is a much more difficult task.

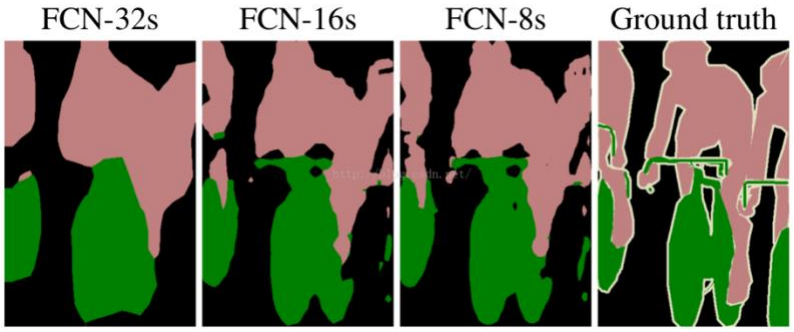


An example of FCN architecture used for semantic segmentation

There are variants of FCN architecture:

1. **FCN-32**: Directly produces the segmentation map from *conv7*, by using a transposed convolution layer with **stride 32**, we build FCN-32 with the same size of input image. But it also makes the output label map rough. This is because deep features can be obtained when going deeper, spatial location information is also lost when going deeper. That means output from shallower layers have more location information. If we combine both through so called skip structure, we will enhance the result. We fuse output by element-wise addition.
2. **FCN-16**: Sums the 2x upsampled prediction from *conv7* (using a transposed convolution layer with stride 2) with pool4 and then produces the segmentation map, by using a transposed convolutional layer with **stride 16** on top of that.
3. **FCN-8**: Sums the 2x upsampled *conv7* (with a stride 2 transposed convolution) with *pool4*, upsamples them with a stride 2 transposed convolution and sums them with *pool3*, and applies a transposed convolution layer with **stride 8** on the resulting feature maps to obtain the segmentation. This fusing operation actually is like the boosting or ensemble technique used in other famous CNN architectures like AlexNet, VGGNet, and GoogLeNet, where they add the results by multiple model to make the prediction more accurate. But in this case, it is done for each pixel, and they are added from results of different layers within a model.

As explained above, the upsampling paths of the FCN variants are different, since they use different skip connection layers and strides for the last convolution, yielding different segmentations, as shown in figure below. Combining layers that have different precision helps retrieving fine-grained spatial information, as well as coarse contextual information.



CSC 449 Homework Four
Yangxin Fan

References:

1. https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1
2. Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
3. http://deeplearning.net/tutorial/fcn_2D_segm.html