Fully Convolutional Networks for Semantic Segmentation

Comment

IU (Intersection of Union)

In object detection, we define ***IU(IoU)*** between actual bounding box and the one estimated by model as following,

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Abstract

- "fully convolutional nteworks" taking input of arbitrary size and produce corresondingly-sized output with efficient inference and learning.
- adapting contemporary networks into FCN and transfer their learned representations by fine-tuning
- < 0.2 sec for a typical image

1. Introduction

- Train FCNs end-to-end **for pixelwise prediction** and **from supervised pre-training**
- In-network upsampling layers enable pixelwise predictio nand learninig in nets with subsample dpooling
- Defined skip architecture that combines deep, coarse, semantic information and shallow, fine, appearance information.

2. Related work

- The paper draws on recent successes of dep nets for image classification and transfer learning
- Fully convolutional network
- Dense prediction with convnets

It adapts and extend deep classification architectures, using image classification as supervised pre-training, and fine-tune fully convolutoinally to learn simply and efficiently from whole image inputs and ground truths. Also, they fuse features across layers to define a nonlinear local-to-global representation

3. FCN

Each layer of data in a convnet is $h \times w \times d$

$$y_{ij} = f_{ks}(\{\mathbf{x}_{si+\delta i, sj+\delta j}\}_{0 \leq \delta i, \delta j \leq k})$$

where k is the kernel size, s is the stride or subsampling factor, and f_{ks} determines the layer type – convolution or pooling, activation function Also,

$$f_{ks} \circ g_{k's'} = (f \circ g)_{k'+(k-1)s', ss'}$$

Since

$$l(x; \theta) = \sum_{ij} l'(\mathbf{x}_{ij}; \theta)$$

SGD on l computed on whole images will be the same as SGD on l', taking all of the final layer receptive fields as a minibatch

Although the receptive fields overlap significantly, the computation of both feed-forward *and* backpropagation are much more efficient when computed layer-by-layer thanks to the inherent computational efficiency and aggressive optimization

3.1 Adapting classifiers for dense prediction

- 5 times faster than the naive approach

3.2 Shift-and-stitch is filter rarefaction

There is a well-known trick for efficiently producing identical results, from wavelet community, although the cost by a factor of f^2, To upsample,

$$f'_{ij} = \begin{cases} f_{i/s, j/s} & \text{if } s \text{ divides both } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

Tradeoff

- Decreasing subsampling – the filters see finer infomration but have smaller receptive fields and take longer to compute
- Shift-and-stitch – the output is denser without decreasing the receptive field sizes of the filters, but the filters are prohibited from accessing information at a finer scale

3.3 Upsampling is backwards strided convolution

- In a sense of interpolation, upsampling with factor f is convolution with a fractional input stride of $1/f$.

3.4 Patchwise training is loss sampling

(further study needed)

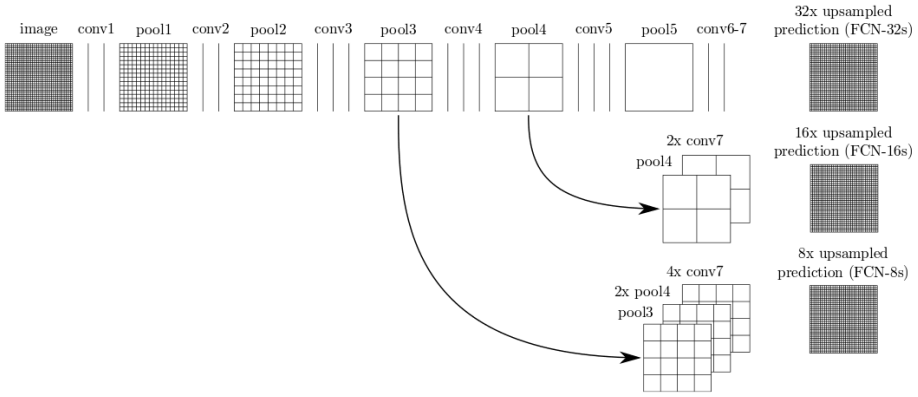
4. Segmentation Architecture

It trains for segmentation by fine-tuning, and then add skips between layers to fuse coarse, semantic and local, appearance information.

4.1 From classifier to dense FCN

- They decapitate each net by discarding the final classifier layer, and convert all FCNs to convolutions and append a 1×1 convolution with channel dimension 21 to predict scores

4.2 Combining what and where



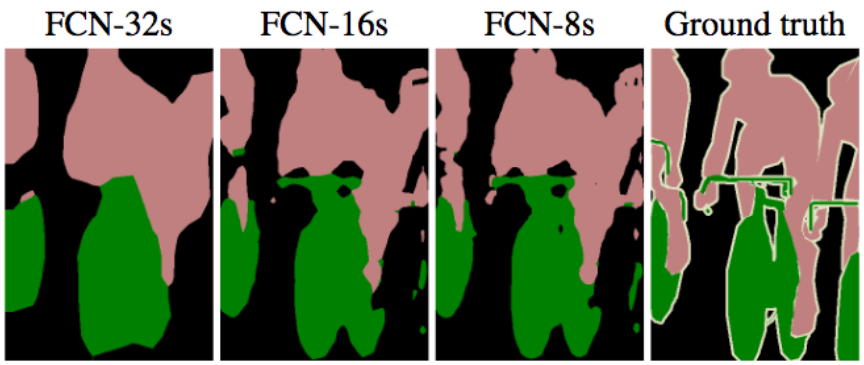


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

4.3 Experimental framework

5. Results

- mean IU - R-CNN 4.79 / FCN-8s 62.7

6. Conclusion