Segmentation

Suggested reading:

A short guide to semantic segmentation

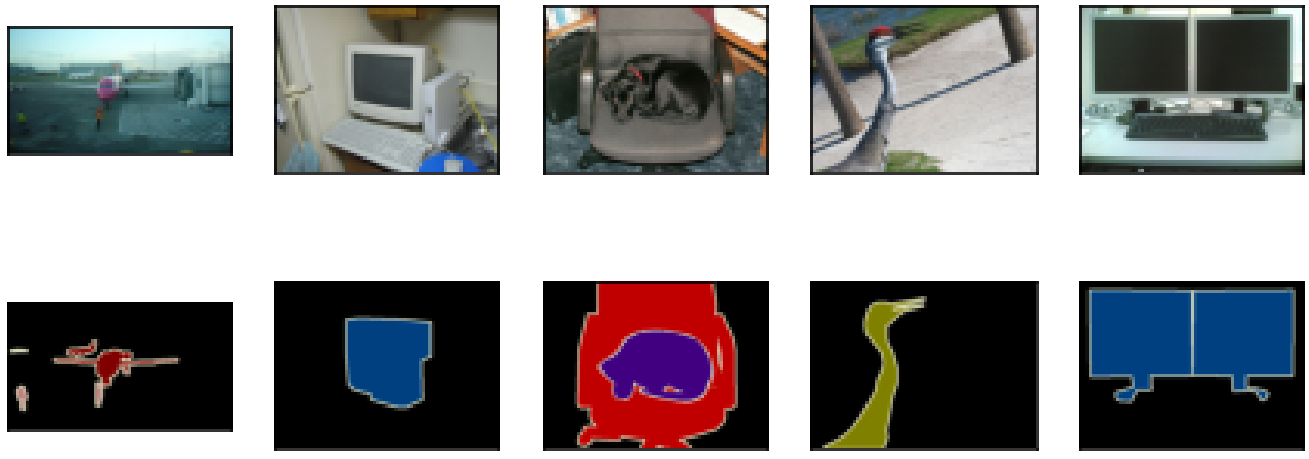
Semantic Segmentation

Semantic segmentation: classify each pixel in an image according to the object category it belongs to.

Building supervised training set is expensive, since it requires a complex human operation.

Semantic Segmentation

The label is itself an image, with a different color for each category.



For this reason, semantic segmentation can be regarded as a special case of Image-to-image transformation

Pascal VOC 2012

PASCAL Visual Object Classes

20 classes:

- Person: person
- Animal: bird, cat, cow, dog, horse, sheep
- Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
- Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.

Cityscapes dataset

Semantic understanding of urban street scenarios

A large-scale dataset containing stereo video sequences recorded in street scenes from 50 different cities, with high quality pixel-level annotations of 5K frames, and a larger set of 20K weakly annotated frames.

More datasets

- Syntia a collection of photo-realistic frames rendered from a **virtual** city and **precise** pixel-level semantic annotations for 13 classes: misc, sky, building, road, sidewalk, fence, vegetation, pole, car, sign, pedestrian, cyclist, lane-marking.
- ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. ScanNet is an RGB-D video dataset containing 2.5 million views in more than 1500 scans, annotated with 3D camera poses, surface reconstructions, and instance-level semantic segmentations.
- Sun RGB-D: 10,000 RGB-D images of densely annotated indoor scenes, includes 146617 2D polygons and 58657 3D bounding boxes with accurate object orientations, as well as a 3D room layout and category for scenes.

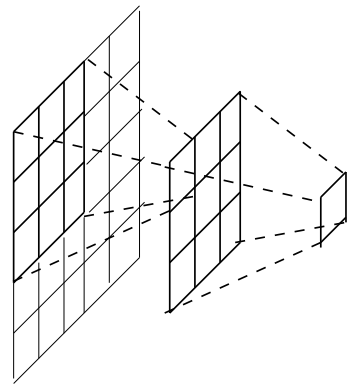
Convolutionalization

Suggested reading:
Fully Convolutional Networks for Semantic Segmentation. E.Shelhamer, J.Long, T.Darrell.

Composition of convolutions

Composing convolutions we still get a convolution

Specifically, the composition of convolutional layers essentially behaves as a convolutional layer.



The stride of the compund covolution is the product of the strides of the components.

What about the kernel dimension?

Dimension of compound kernel

Remember that, neglecting padding,

$$\frac{D_{in} - K}{S} + 1 = D_{out}$$

or equivalently

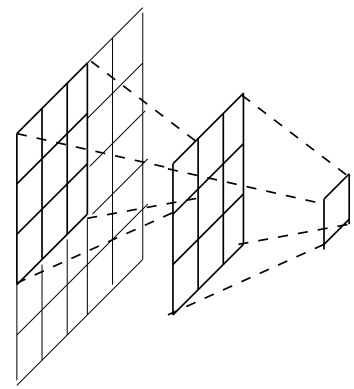
$$D_{in} = S * (D_{out} - 1) + K$$

Suppose to compose two kernels with dimension 3 and stride 1. Then, the intermediate dimension is

$$(1 - 1) * 1 + 3 = 3$$

and the initial dimension must be

$$(3 - 1) * 1 + 3 = 5$$



Another example

Suppose to compose a kernel of dimension $K_1 = 5$ and stride $S_1 = 3$ with another kernel of dimension $K_2 = 3$ and stride $S_2 = 2$.
Then, applying the rule

$$D_{out} = S * (D_{out} - 1) + K$$

we get, for $D_2 = 1$,

$$D_1 = S_2 * (D_2 - 1) + K_2 = 3$$

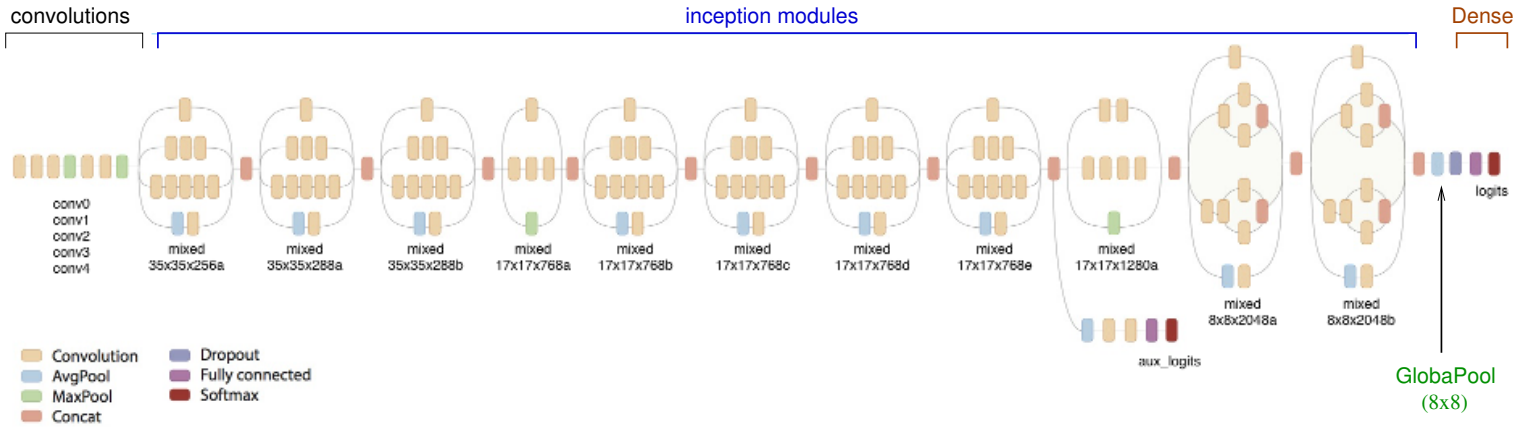
and

$$D_0 = S_1 * (D_1 - 1) + K_1 = 11$$

This is the dimension of the compund kernel, aka its "receptive fields".

What breaks convolutionality

Let us consider a typical architecture for image classification



Composing convolutional layers we still get a convolutional network.

What breaks convolutionality are the dense layers at the end of networks (if maxpooling has a fixed pooling dimension).

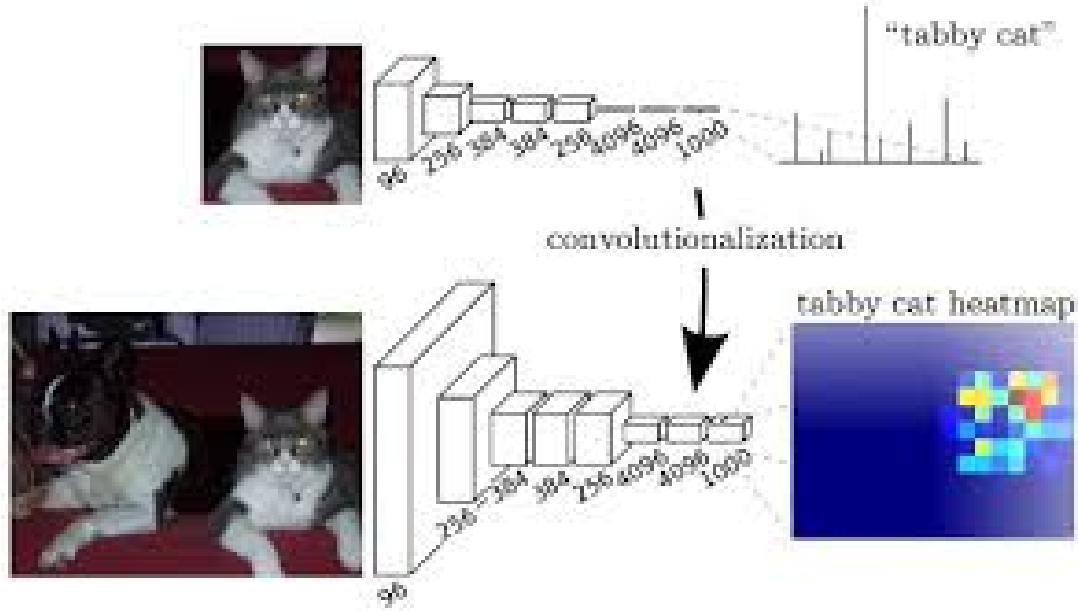
A simple idea

- neurons in dense or convolutional layers share the same functionalities: they simply compute weighted sums of their inputs (dot products)
- we can look at a dense layer as a convolution with a particularly large filter (equal to its input size)
- this way each dense layer can be turned into a convolutional layer (reusing the same weights), making the resulting networks fully convolutional.

What we get

If we start from an image classification network working on images of a given, fixed size, we get a DCNs that

- takes in input images of arbitrary dimension
- produces in output a heatmap of activations of the different objects categories, relative to different locations of the input image upon which the DCN was convolved.



InceptionV3 as a single convolution

We can look at the whole convnet as a **single convolution**.
What is the stride and what is the kernel size?

The stride is the product of all strides, that is $2^5 = 32$.
For the kernel size we get:

$$\begin{aligned} D_{in} &= ((((D_{out} - 1) * 2 + 2) * 2 + 2) * 2 + 4) * 2 + 4) * 2 + 3 \\ &= 32 D_{out} + 43 \end{aligned}$$

For $D_{out} = 1$, $D_{in} = 32 + 43 = \mathbf{75}$, that is the "**kernel size**", and also the minimal input dimension for inceptionV3.

For $D_{out} = 8$ (dimension of the global pooling layer), we get $D_{in} = 32 * 8 + 43 = 256 + 43 = \mathbf{299}$ that is the "canonical" input dimension for inceptionV3.

See Automatic point-of-interest image cropping via ensembled convolutionalization for similar computations for other networks.

Big stride = rough localization

InceptionV3, can be seen as a single convolution with

$$\text{Kernel size} = 75, \text{S}tride = 32$$

Given an image with input dimension D_{in}, the dimension of the resulting heatmap will be

$$\frac{D_{in} - 75}{32} + 1$$

that is very coarse.

localization of objects is far from precise!

Where and what

The 32 pixel stride at the final prediction layer limits the scale of detail in the upsampled output.

Shelhamer et al. address this issue by adding skips that combine the final prediction layer with lower layers with finer strides.

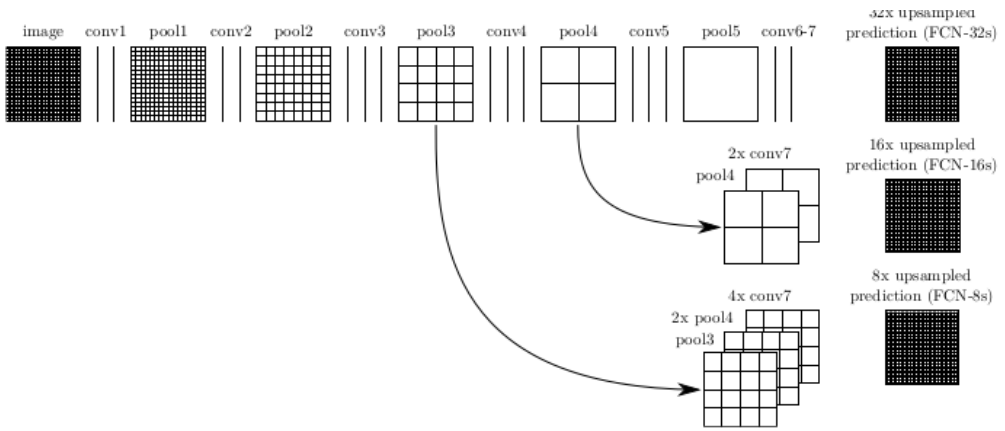


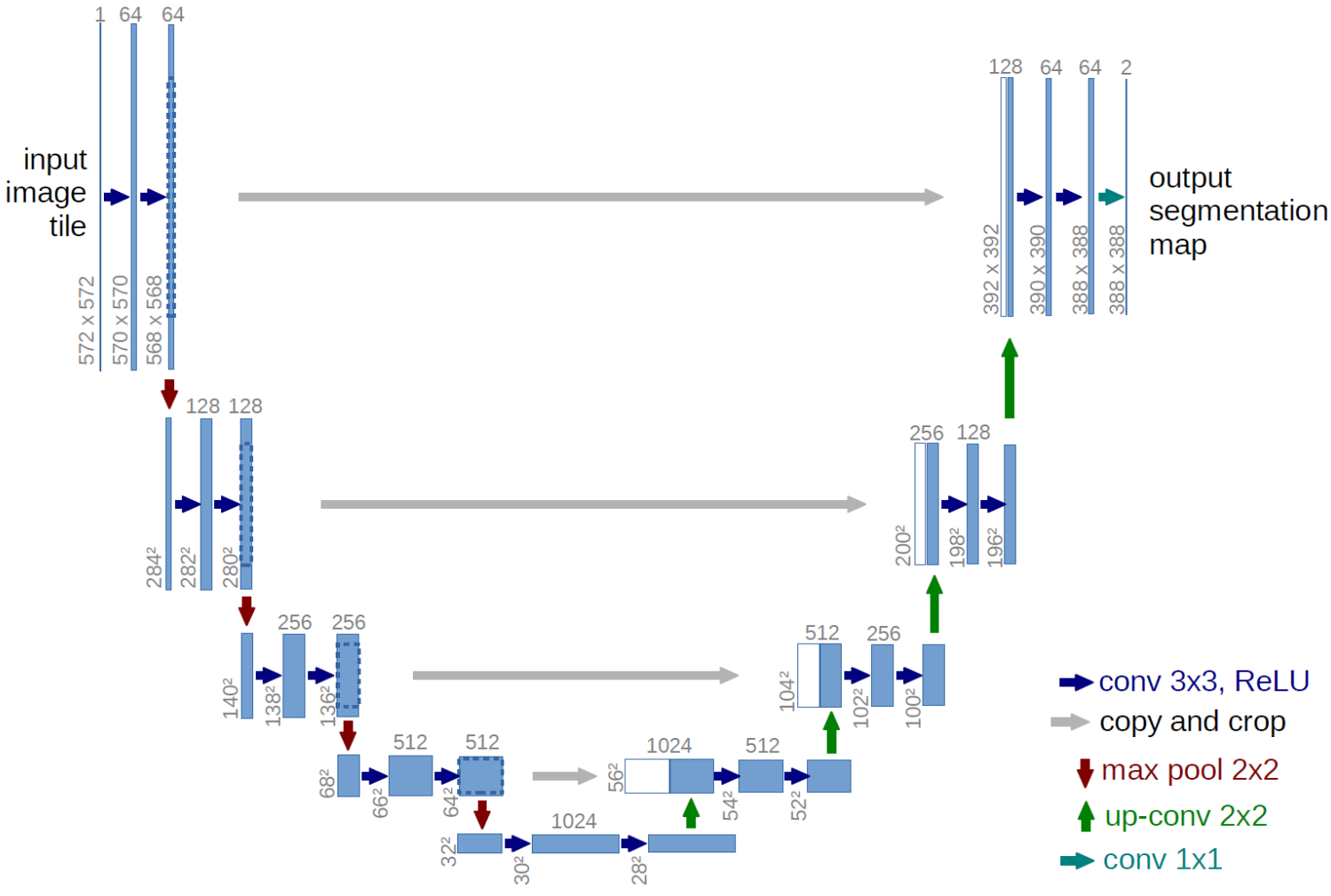
Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

U-net

Suggested reading:

U-Net: Convolutional Networks for Biomedical Image Segmentation. By O.Ronneberger, P.Fischer, T.Brox

U-net


input image tile
572 x 572
570 x 570
568 x 568
1 64 64
128 128
284² 282² 280²
256 256
140² 138² 136²
512 512
68² 66² 64²
1024
32² 30² 28²
1024
512
56² 54² 52²
512 256
104² 102² 100²
256 128
200² 198² 196²
128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

U-net key properties

It does not rely on a classification network.

U-net learns segmentation in an end-to-end setting.

It can work with relatively few training images (data augmentation was largely exploited) and yields accurate precise segmentations.