2014

# Fully Convolutional Network

[Link]

Jonathan Long | Evan Shelhamer | Trevor Darrell

Reading note - Hao Tsui, 08.Apr.2021

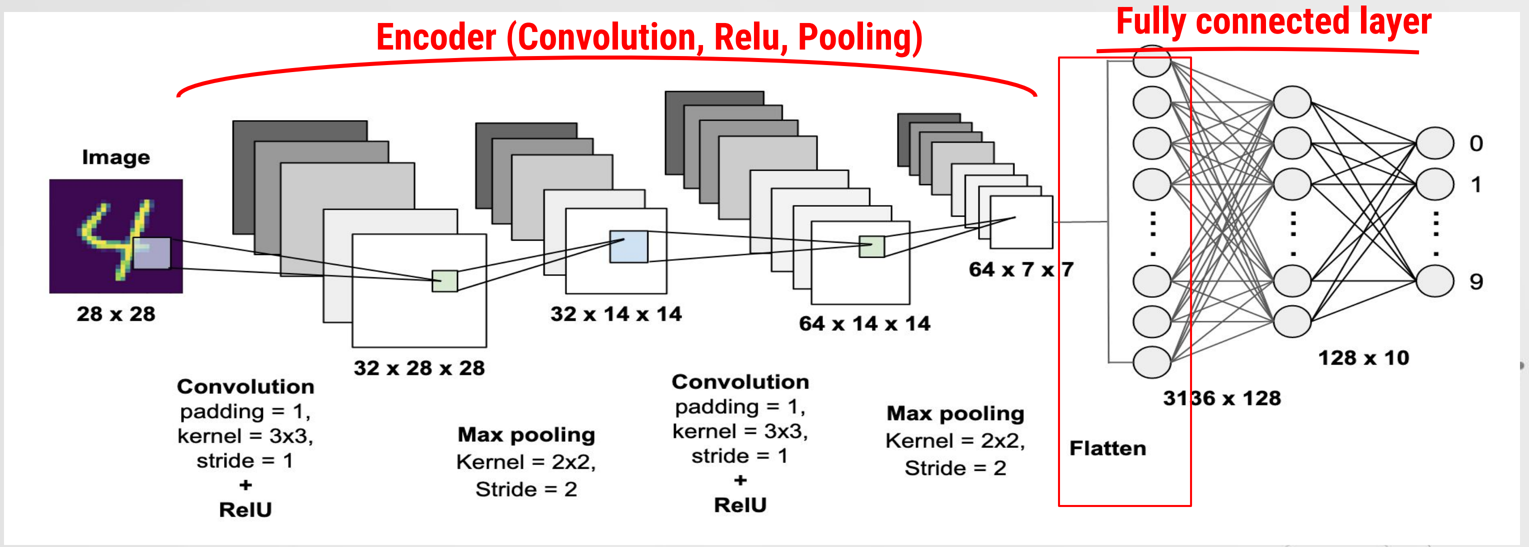
# Fully Convolutional Network (FCN)

1. Architecture
    a. Can take arbitrary size as input
    b. Up sampling - Transposed Convolution/Deconvolution/Fractionally-strided convolution
    c. Skip connection
2. Pixel-wise semantic segmentation

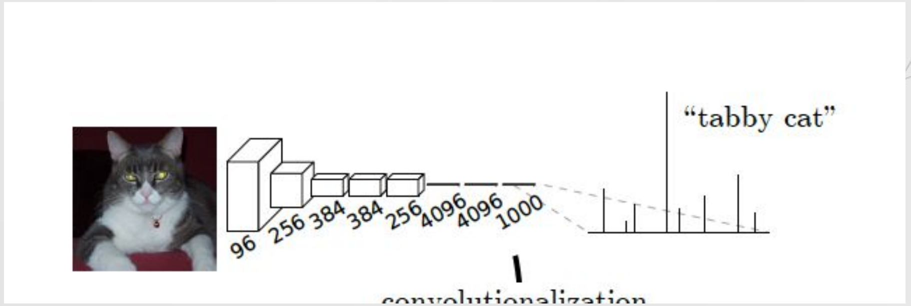
# 01

## Architecture

## Traditional CNN Architecture

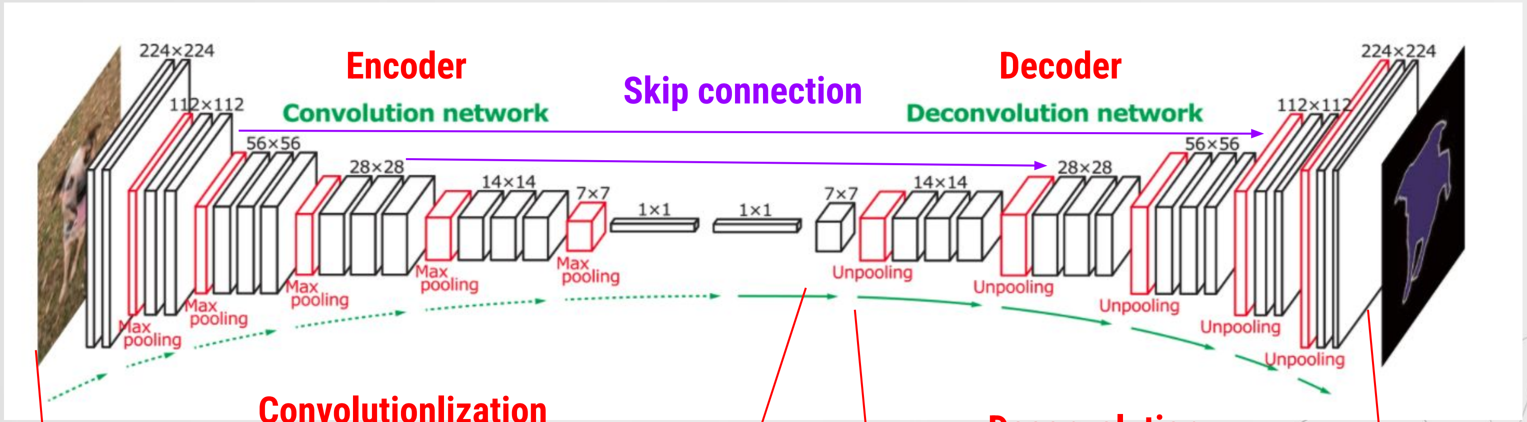


Encoder (Convolution, Relu, Pooling)
Fully connected layer
Image
28 x 28
32 x 28 x 28
32 x 14 x 14
64 x 14 x 14
64 x 7 x 7
3136 x 128
128 x 10
0
1
9
Convolution padding = 1, kernel = 3x3, stride = 1 + RelU
Max pooling Kernel = 2x2, Stride = 2
Convolution padding = 1, kernel = 3x3, stride = 1 + RelU
Max pooling Kernel = 2x2, Stride = 2
Flatten

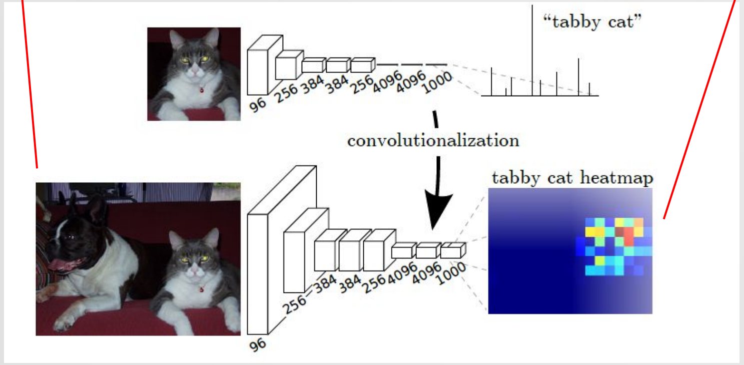


## Fully Convolutional Network

Operates on an input of any size, and produces and output of corresponding spatial dimensions



Convolutionlization



Deconvolution

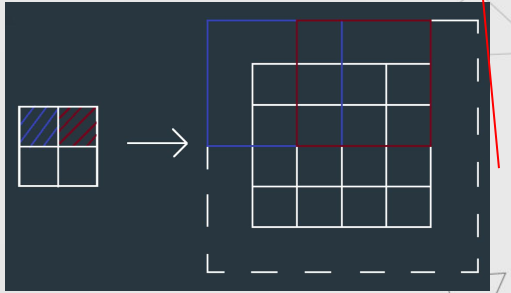
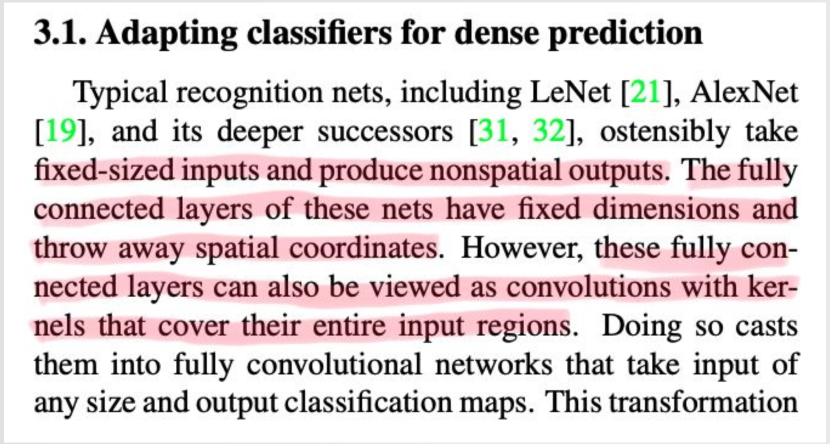


*Image credit: http://cvlab.postech.ac.kr/research/deconvnet/*

# 02

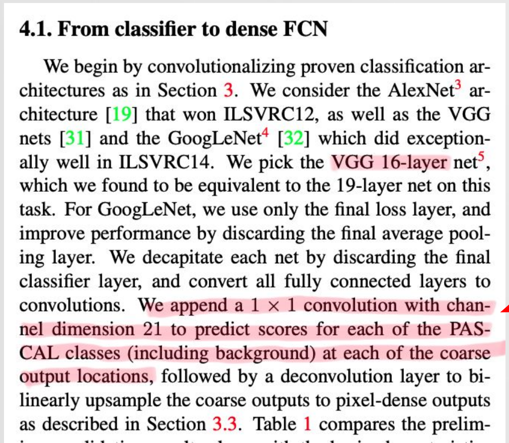
## Convolutionalization

## Fully Convolutional Network



### 3.1. Adapting classifiers for dense prediction

Typical recognition nets, including LeNet [21], AlexNet [19], and its deeper successors [31, 32], ostensibly take fixed-sized inputs and produce nonspatial outputs. The fully connected layers of these nets have fixed dimensions and throw away spatial coordinates. However, these fully connected layers can also be viewed as convolutions with kernels that cover their entire input regions. Doing so casts them into fully convolutional networks that take input of any size and output classification maps. This transformation



### 4.1. From classifier to dense FCN

We begin by convolutionalizing proven classification architectures as in Section 3. We consider the AlexNet[3] architecture [19] that won ILSVRC12, as well as the VGG nets [31] and the GoogLeNet[4] [32] which did exceptionally well in ILSVRC14. We pick the VGG 16-layer net[5], which we found to be equivalent to the 19-layer net on this task. For GoogLeNet, we use only the final loss layer, and improve performance by discarding the final average pooling layer. We decapitate each net by discarding the final classifier layer, and convert all fully connected layers to convolutions. We append a 1 × 1 convolution with channel dimension 21 to predict scores for each of the PASCAL classes (including background) at each of the coarse output locations, followed by a deconvolution layer to bilinearly upsample the coarse outputs to pixel-dense outputs as described in Section 3.3. Table 1 compares the prelim-

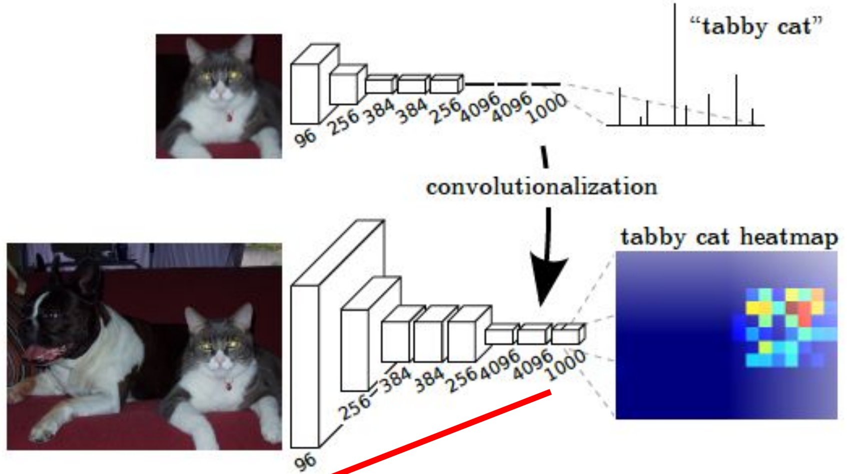


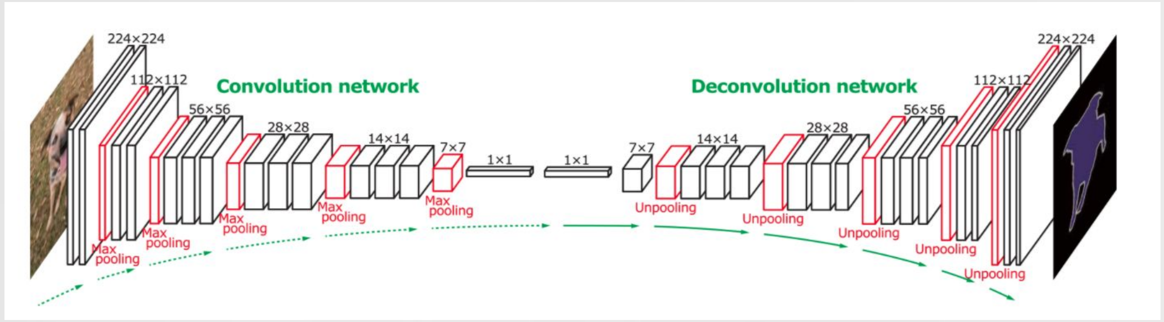
Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

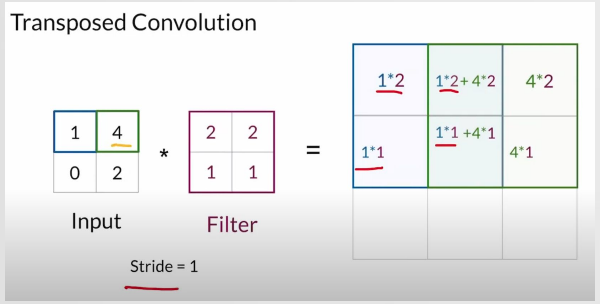
# 03

## Deconvolution

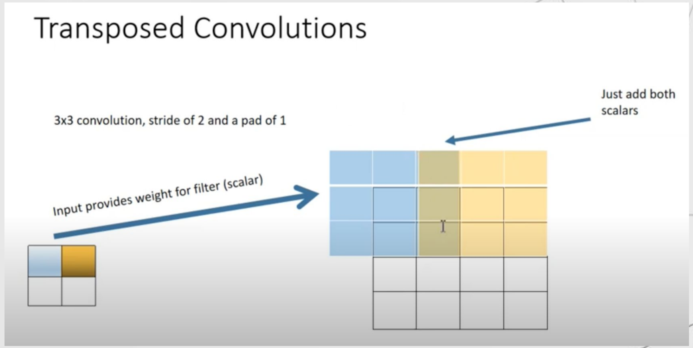
Transposed Convolution

# Deconvolution/Transposed Convolution





GANs Course | deeplearning.ai | Week 2: Transposed Convolutions


Transposed Convolutions
3x3 convolution, stride of 2 and a pad of 1
Input provides weight for filter (scalar)
Just add both scalars

## 04

## Skip Connection

Retain the information from finer to coarser

## Combine coarser and finer information

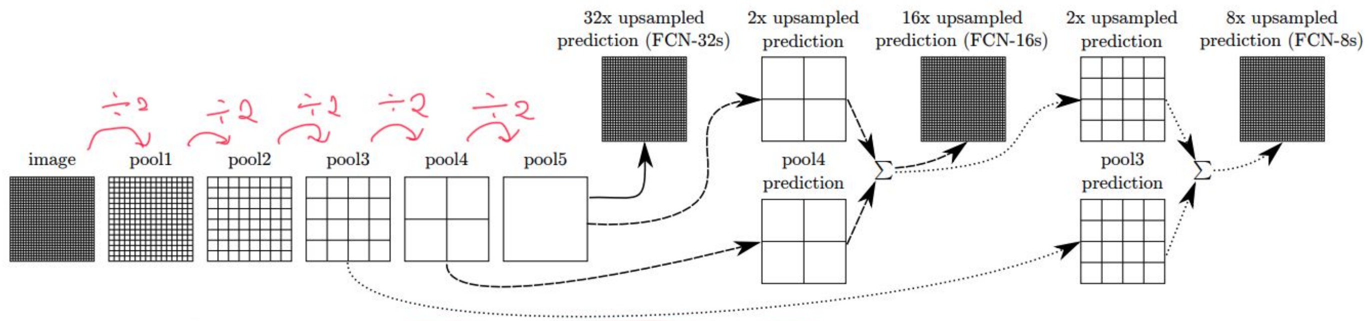
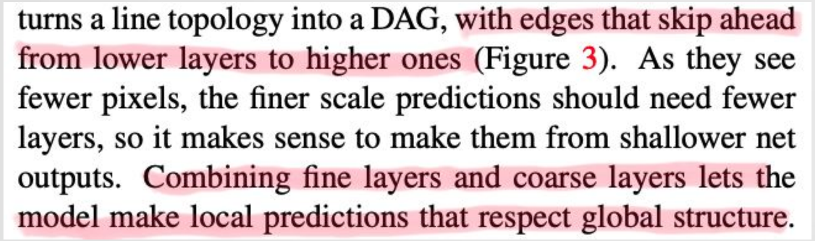


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Layers are shown as grids that reveal relative spatial coarseness. Only pooling and prediction layers are shown; intermediate convolution layers (including our converted fully connected layers) are omitted. Solid line (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Dashed line (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. Dotted line (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.



turns a line topology into a DAG, with edges that skip ahead from lower layers to higher ones (Figure 3). As they see fewer pixels, the finer scale predictions should need fewer layers, so it makes sense to make them from shallower net outputs. Combining fine layers and coarse layers lets the model make local predictions that respect global structure.

05

## Conclusion

## Result

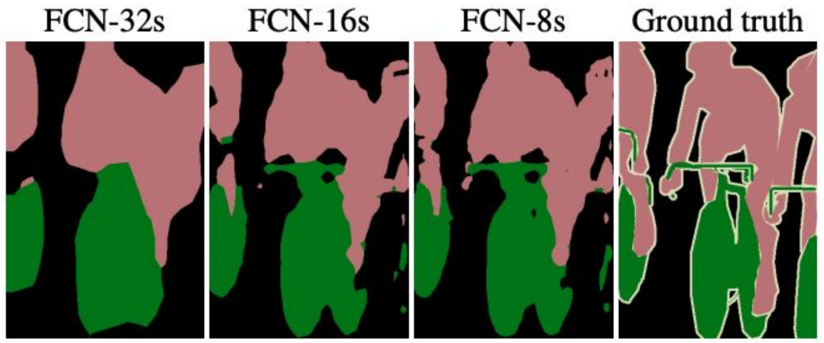


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

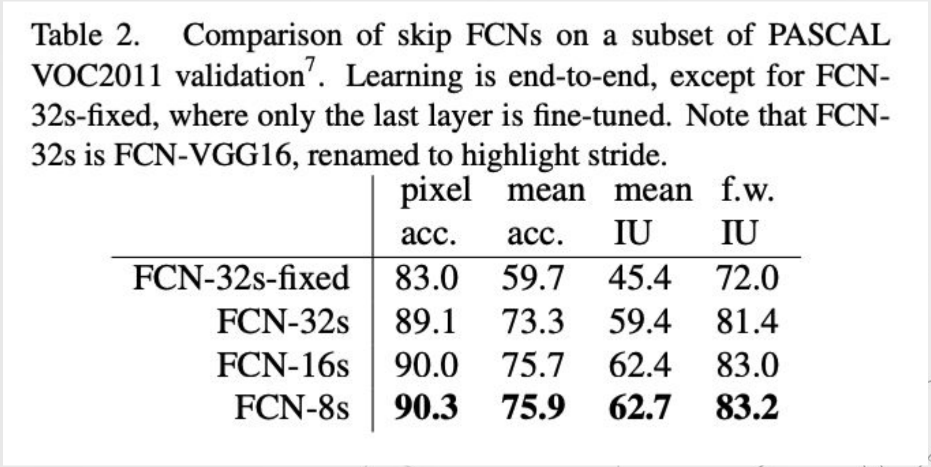


Table 2. Comparison of skip FCNs on a subset of PASCAL VOC2011 validation[7]. Learning is end-to-end, except for FCN-32s-fixed, where only the last layer is fine-tuned. Note that FCN-32s is FCN-VGG16, renamed to highlight stride.

| | pixel acc. | mean acc. | mean IU | f.w. IU |
|---|---|---|---|---|
| FCN-32s-fixed | 83.0 | 59.7 | 45.4 | 72.0 |
| FCN-32s | 89.1 | 73.3 | 59.4 | 81.4 |
| FCN-16s | 90.0 | 75.7 | 62.4 | 83.0 |
| FCN-8s | **90.3** | **75.9** | **62.7** | **83.2** |