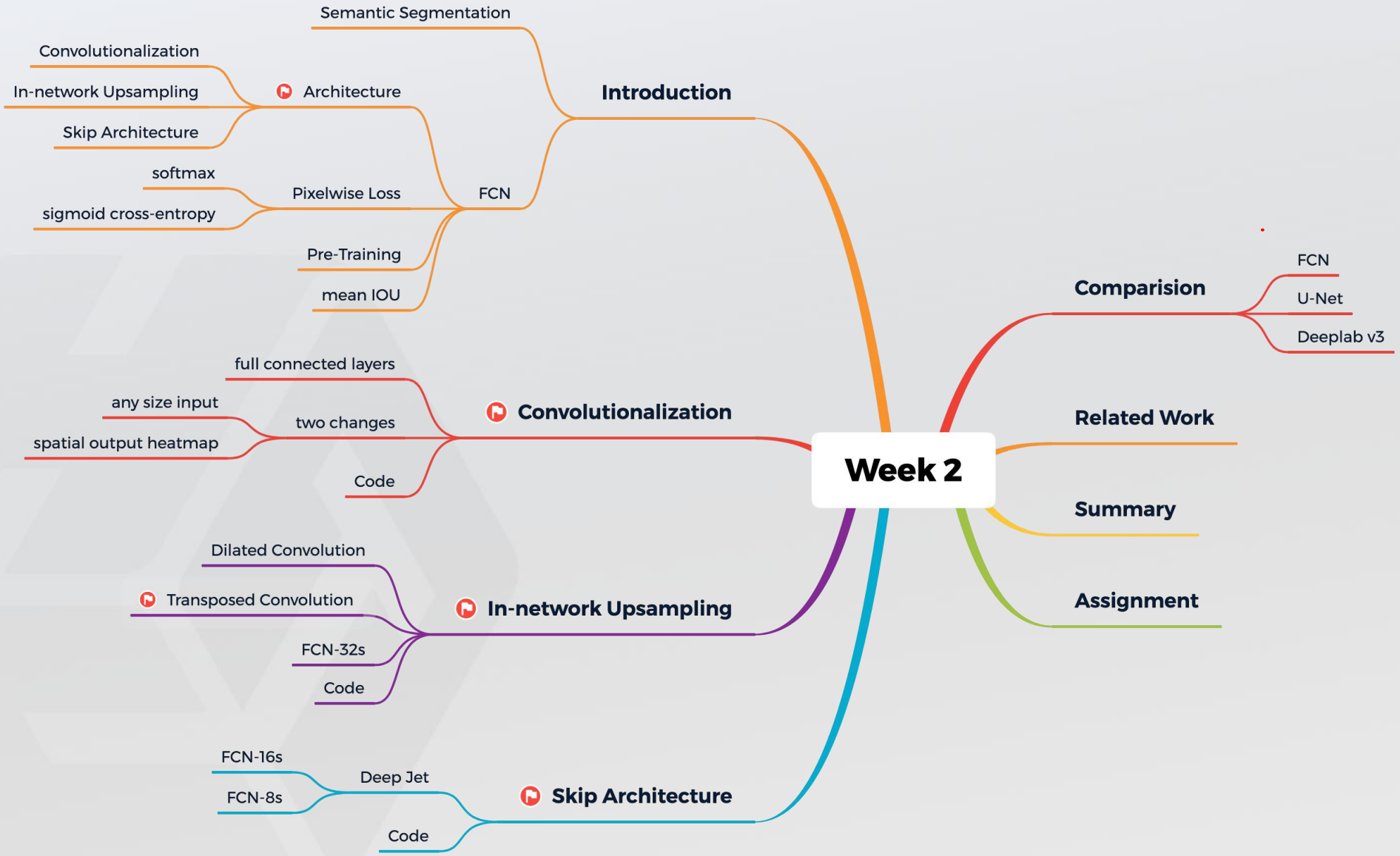
后厂理工学院
houchangtech.com

Lane Segmentation Week 2

HCT CV Class

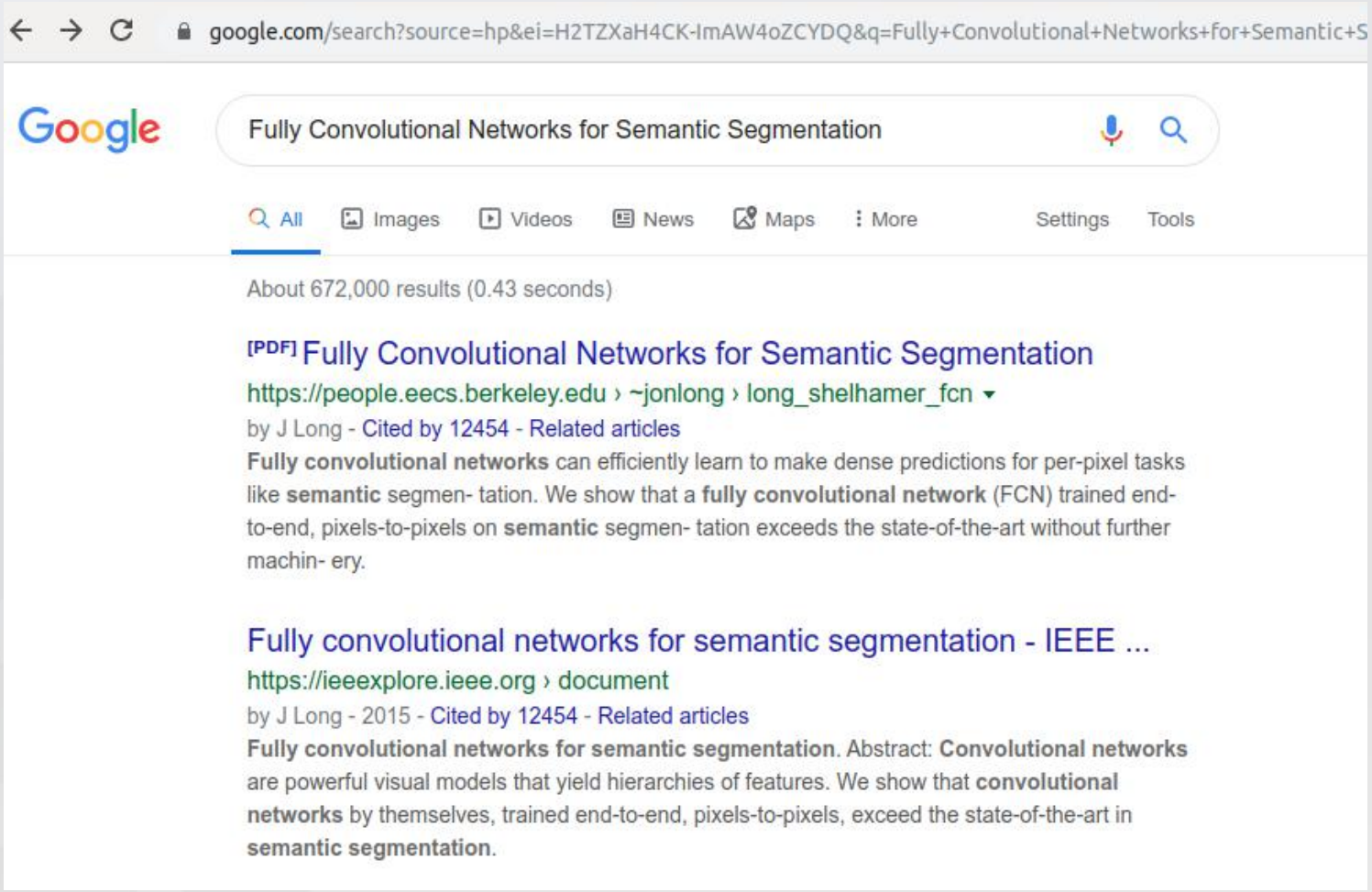
主要内容



学习目标

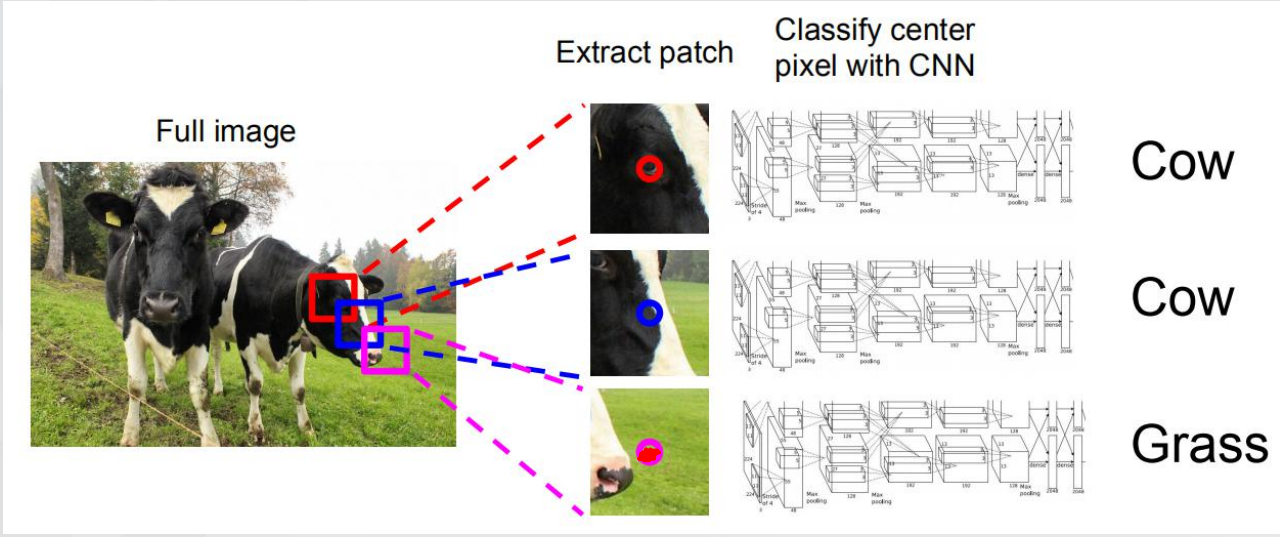
- 理解FCN的原理
- 掌握FCN的关键技术点
- 掌握FCN的实现

FCN


google.com/search?source=hp&ei=H2TZXaH4CK-ImAW4oZCYDQ&q=Fully+Convolutional+Networks+for+Semantic+S
Google
Fully Convolutional Networks for Semantic Segmentation
All
Images
Videos
News
Maps
More
Settings
Tools
About 672,000 results (0.43 seconds)
[PDF] Fully Convolutional Networks for Semantic Segmentation
https://people.eecs.berkeley.edu › ~jonlong › long_shelhamer_fcn
by J Long - Cited by 12454 - Related articles
Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmen- tation. We show that a fully convolutional network (FCN) trained end-to-end, pixels-to-pixels on semantic segmen- tation exceeds the state-of-the-art without further machin- ery.
Fully convolutional networks for semantic segmentation - IEEE ...
https://ieeexplore.ieee.org › document
by J Long - 2015 - Cited by 12454 - Related articles
Fully convolutional networks for semantic segmentation. Abstract: Convolutional networks are powerful visual models that yield hierarchies of features. We show that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, exceed the state-of-the-art in semantic segmentation.

Before FCN

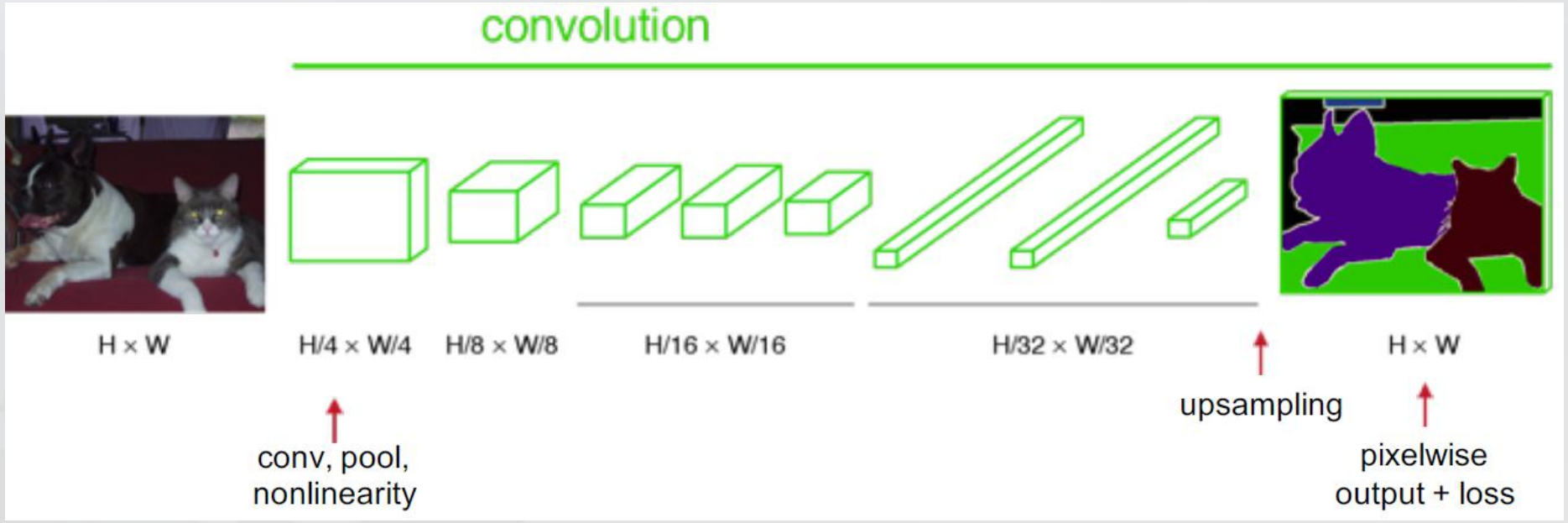
- 使用传统特征
- Sliding Window + Classifier



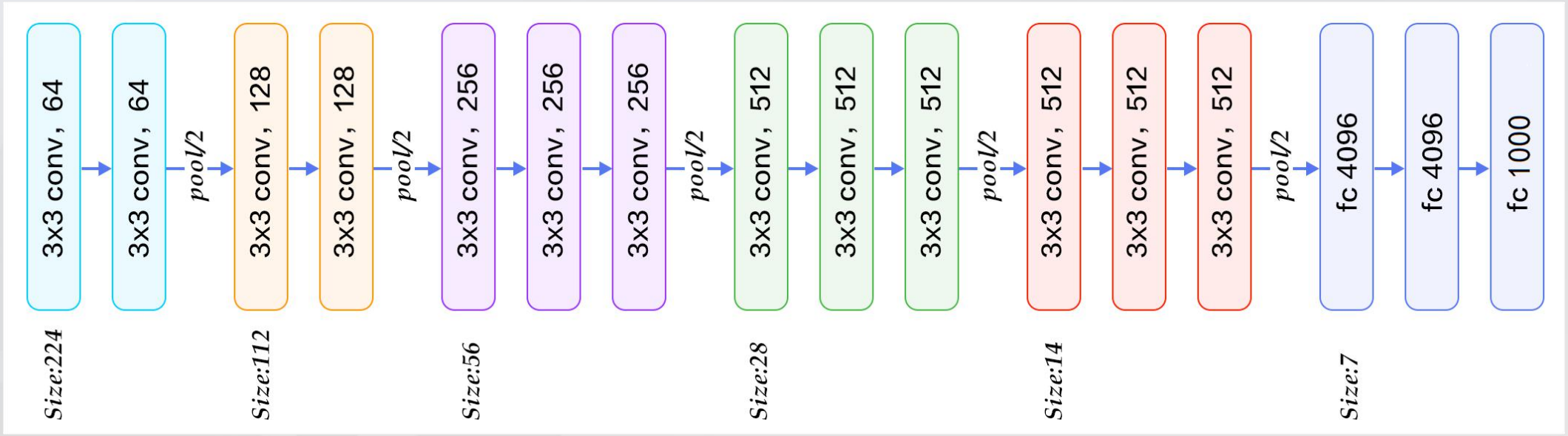
FCN原理

- Fully convolutional versions of existing networks predict dense outputs from arbitrary-sized inputs. Both learning and inference are performed whole-image-at-a-time by dense feedforward computation and backpropagation. In-network upsampling layers enable pixelwise prediction and learning in nets with subsampled pooling.

Convolutionalization



VGG16



VGG16

- nn.Conv2d(3, 64, 3, padding=100)
- nn.ReLU(inplace=True)
- nn.MaxPool2d(2, stride=2, ceil_mode=True)
- nn.Dropout()

VGG16


```
import torch


class VGG16(torch.nn.Module):
        def __init__(self, n_class=21):
                pass


        def forward(self, x):
                pass
```

Classify CNN

- fix-size input
- non-spatial output

卷积层 as 全连接层

- 转置卷积

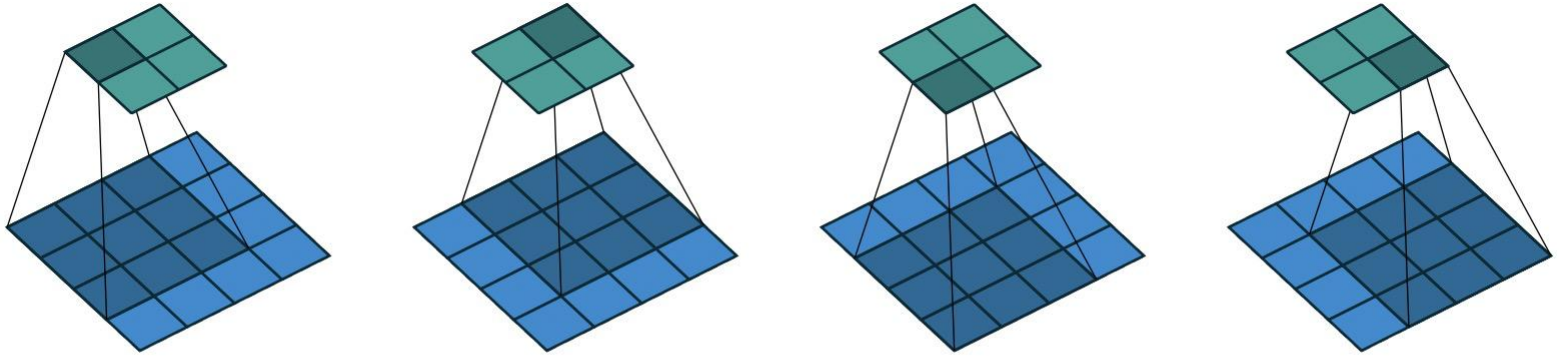


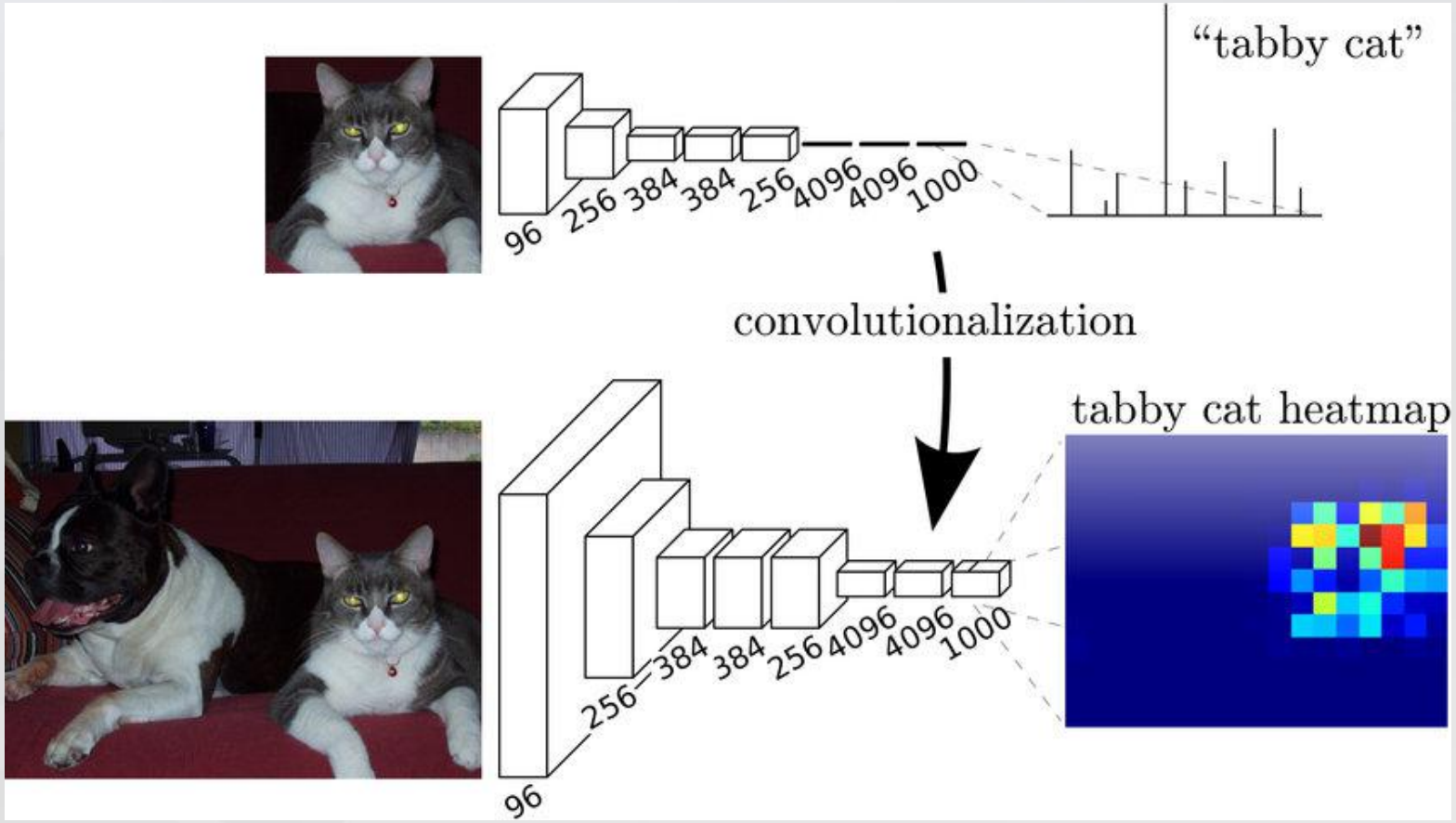
Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

$$\begin{pmatrix} w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} \end{pmatrix}$$

全连接层 as 卷积层

- 全卷积(full sized kernel)

Convolutionalization



Full Convolutional Network

- a net with only layers of this form computes a nonlinear filter, which we call a deep filter or fully convolutional network

- A real-valued loss function composed with an FCN defines a task

VGG16-Conv


```
import torch


class VGG16Conv(torch.nn.Module):
        def __init__(self, n_class=21):
                pass


        def forward(self, x):
                pass
```

VGG16-Conv

Conv1： padding = 100

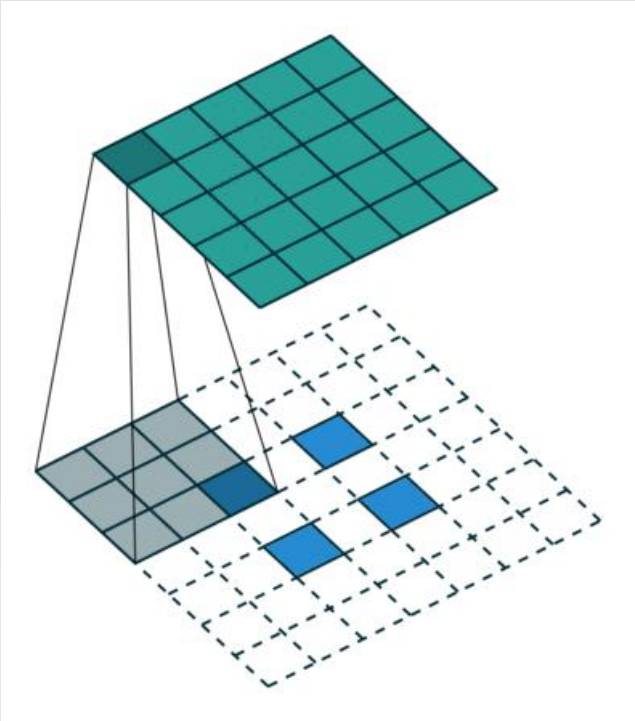
$o = [\ (i + 2p - k)/s\ ] + 1$

VGG16-Conv

- 全卷积结束的位置

In-network Upsampling

- connect coarse outputs to dense pixels



In-network Upsampling

- connect coarse outputs to dense pixels

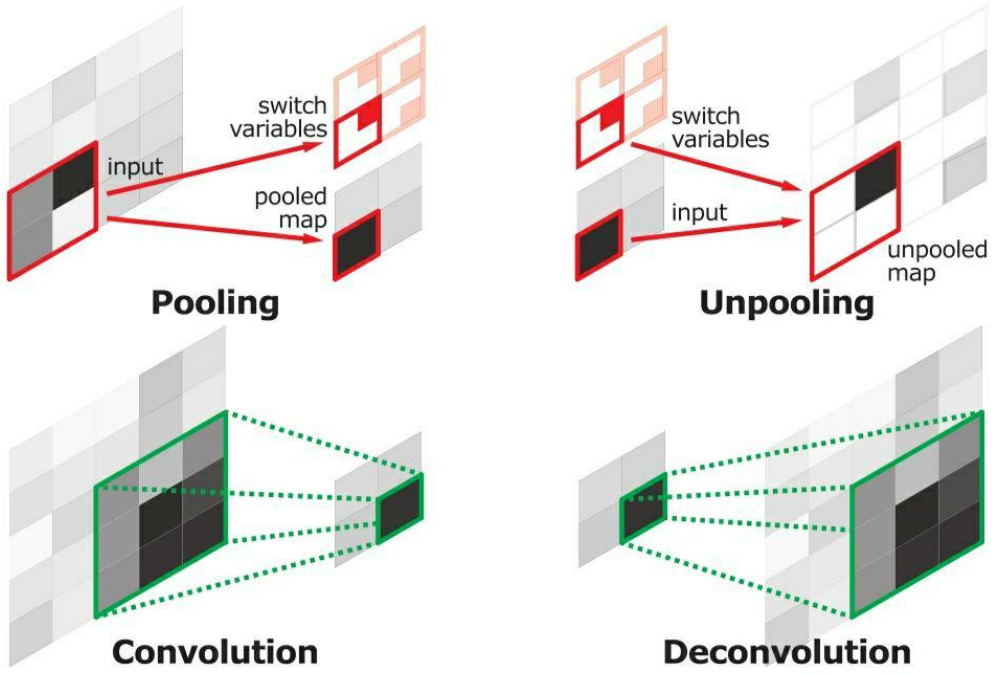


Figure 3. Illustration of deconvolution and unpooling operations.

Transposed Convolution

“A guide to convolution arithmetic for deep learning”

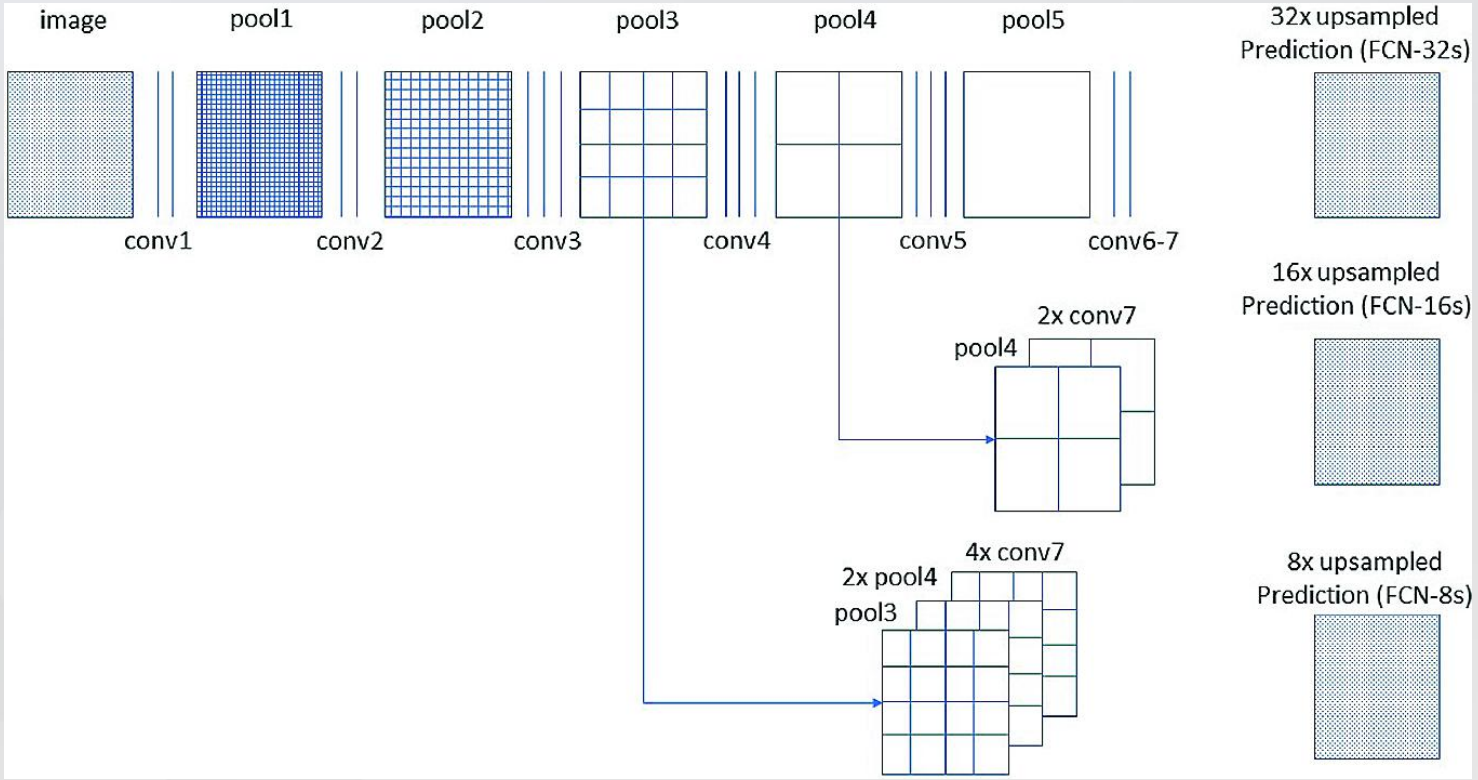
Transposed Convolution

- class torch.nn.**ConvTranspose2d**(in_channels, out_channels, kernel_size, stride=1, padding=0, output_padding=0, bias=True)

- output_padding?

- Set Parameters for upsampling N times

FCN-32s



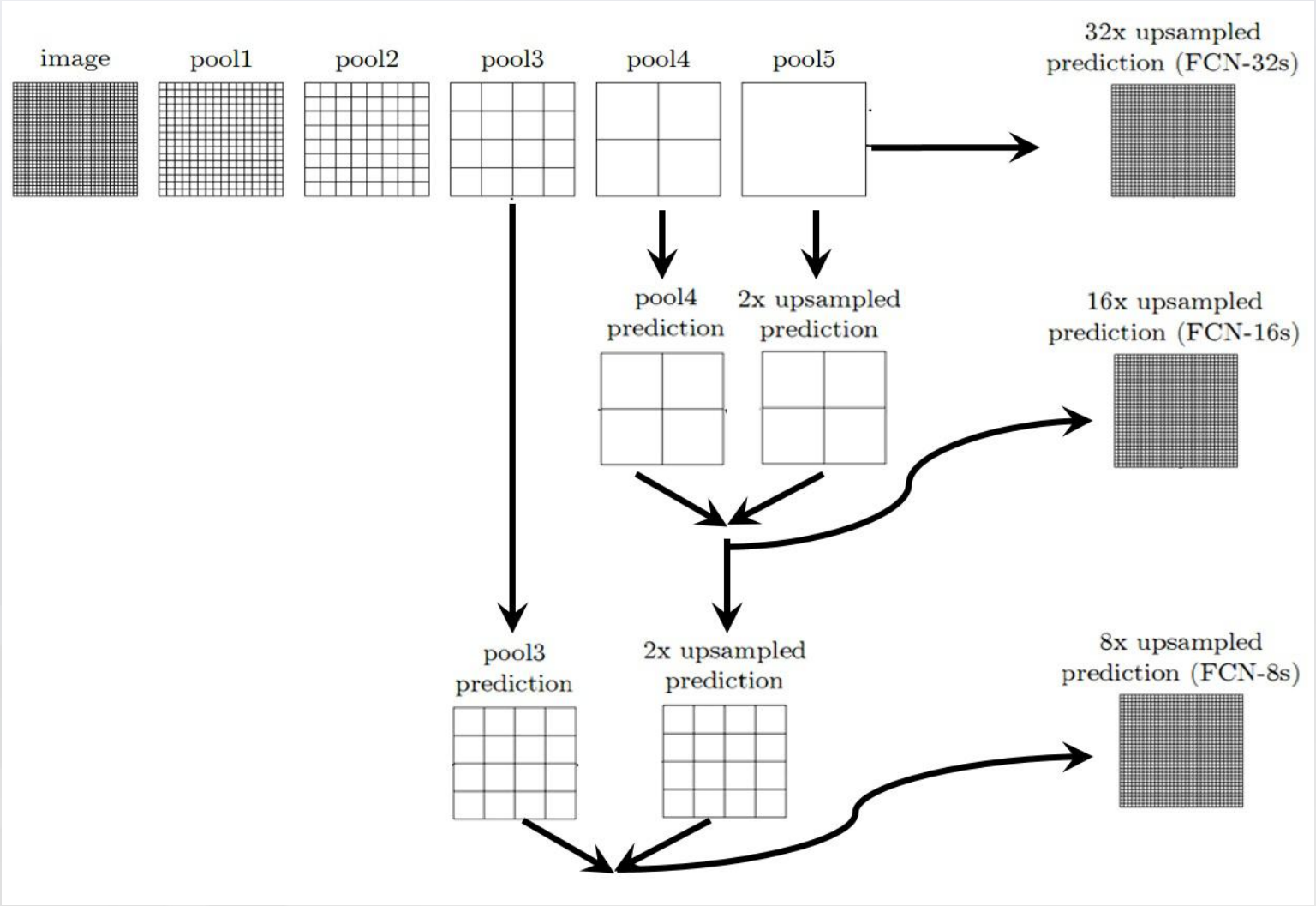
FCN-32s

```
import torch


class FCN32s(torch.nn.Module):
        def __init__(self, n_class=21):
                pass


        def forward(self, x):
                pass
```

Skip Architecture



Skip Architecture

- 维度一致才能相加：spatial/channel
- 1x1 Convolution

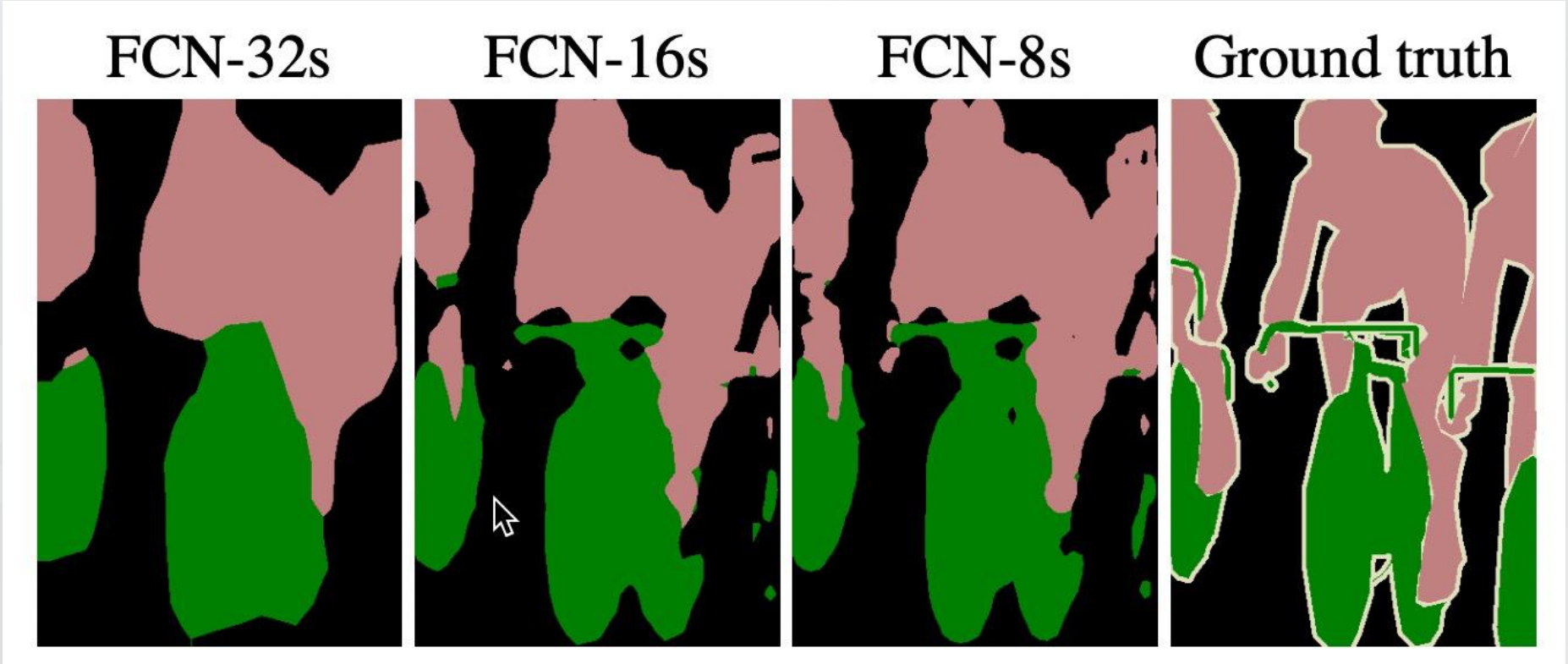
FCN-16s

```
import torch


class FCN16s(torch.nn.Module):
        def __init__(self, n_class=21):
                pass


        def forward(self, x):
                pass
```

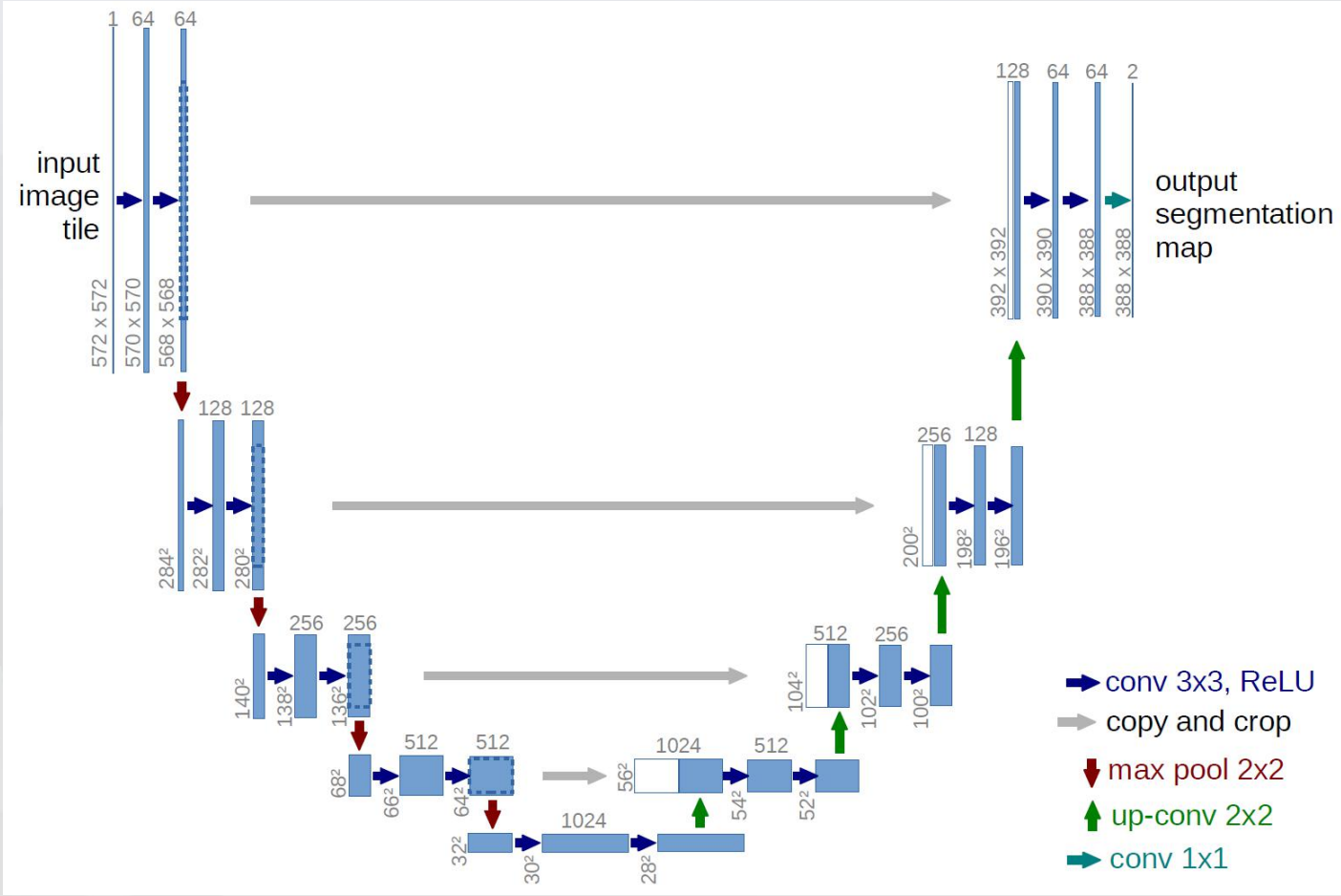
FCN的效果



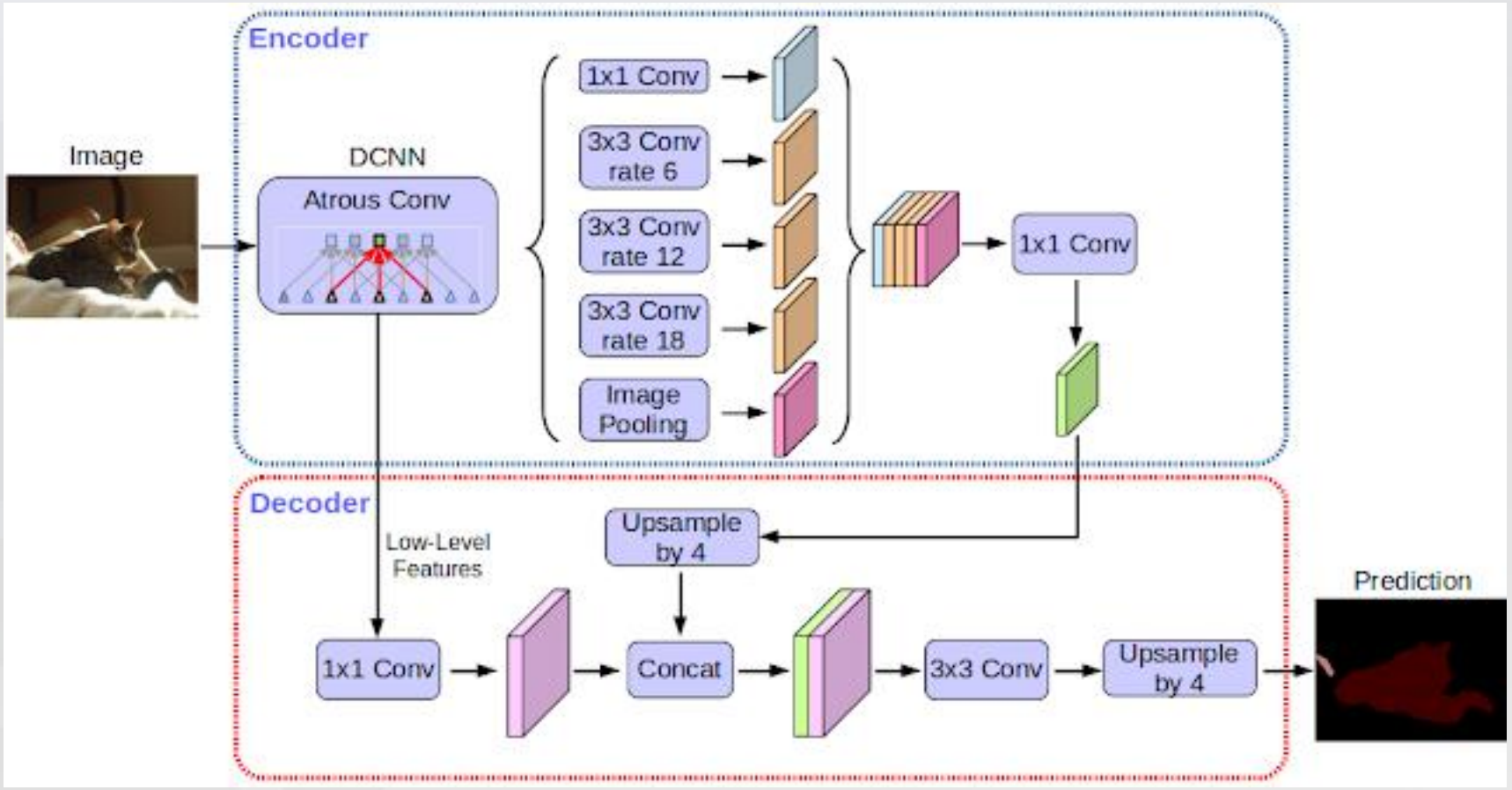
FCN的缺点

- FCN-8s结果还是不够精细
- 没有充分考虑像素之间的关系

模型对比



模型对比


Encoder
Image
DCNN
Atrous Conv
1x1 Conv
3x3 Conv rate 6
3x3 Conv rate 12
3x3 Conv rate 18
Image Pooling
1x1 Conv
Decoder
Low-Level Features
Upsample by 4
1x1 Conv
Concat
3x3 Conv
Upsample by 4
Prediction

课程总结

- 理解FCN的原理
- 掌握CNN分类网络的Convolutionalization方法
- 掌握In-Network Upsampling方法
- 掌握Skip Architecture提升准确度的方法

重难点

- VGG16全卷积化的实现方法
- Transposed Convolution(会计算)
- Skip Architecture(维度一致才能相加)

课程作业

- 实现FCN-8s(VGG16)
- 实现FCN-8s(ResNet101)

课程作业

```
import torch


class FCN8s(torch.nn.Module):
        def __init__(self, n_class=21):
                pass


        def forward(self, x):
                pass
```

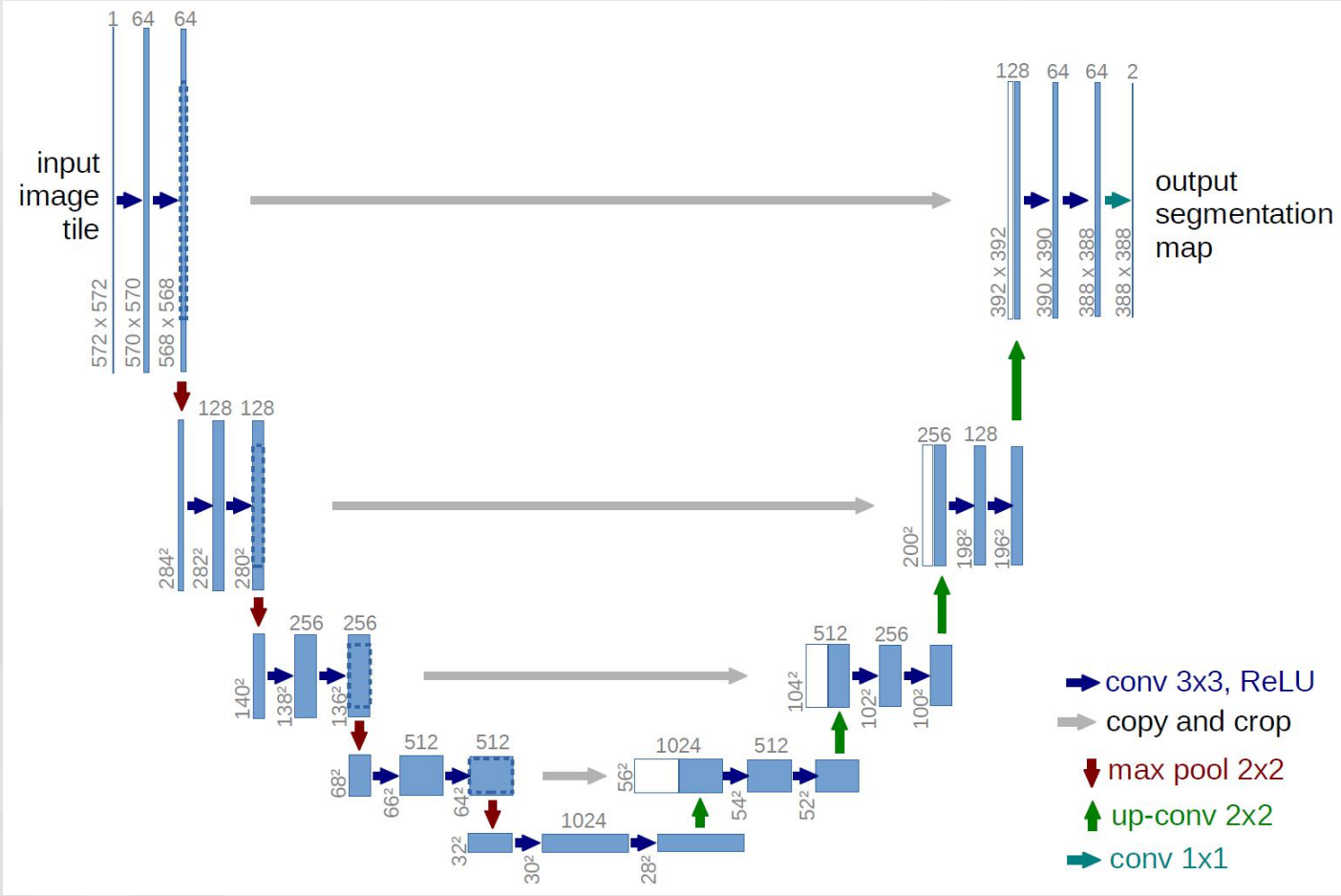
参考资料

- Fully Convolutional Networks for Semantic Segmentation
  https://arxiv.org/abs/1411.4038

- PyTorch Implementation of Fully Convolutional Networks
  https://github.com/wkentaro/pytorch-fcn

- Caffe Reference Implementation of Fully Convolutional Networks
  https://github.com/shelhamer/fcn.berkeleyvision.org

Next Week



后厂理工学院

houchangtech.com

一所专注前沿互联网技术领域的创新实战大学