

AutoEncoders and VAEs

COMP 646: Deep Learning for Vision and Language



RICE UNIVERSITY

Final Project

- I am still giving feedback on the project proposal. There's no grade for the project proposal. There's also no grade for class attendance. See course syllabus. I won't change course syllabus in the middle of the course. I aim to send feedback to everyone before Spring Break ends (hopefully much earlier).
- I will provide you an opportunity to provide a Project Progress Report so you have an idea of my expectations and grade and avoid "surprises".
- Final project deliverables is worth 40% of the class divided as follows:
 - Originality: 10pts
 - Technical accuracy: 10pts
 - Presentation: 10pts
 - Results: 10pts

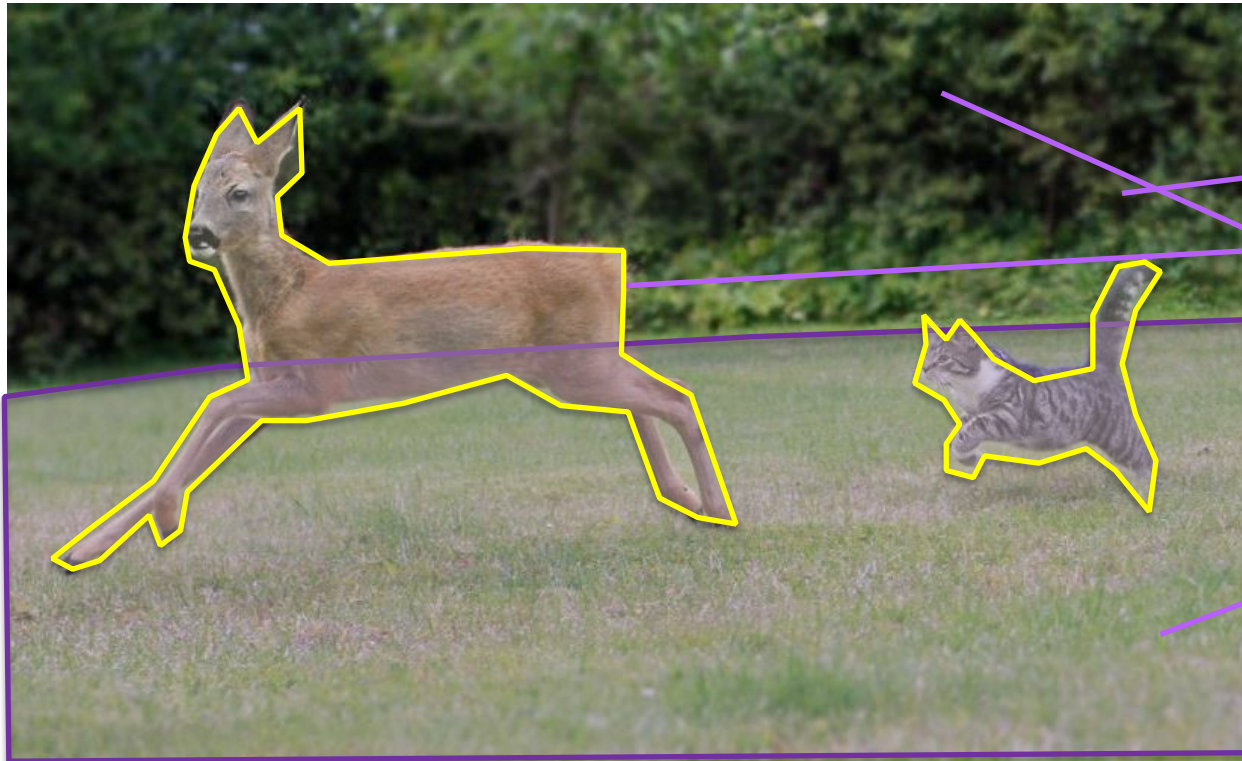
Final Project

- Originality: Is the idea of the project original itself? Is the original idea of the project standard but is any of the experiments original? AI is not very good at providing original ideas but is getting better, 1 out of 3 ideas are good these days. [10pts]
- Technical accuracy: Is your final project report accurate? Are all the terms used appropriately and are the mathematical expressions used to explain your model/loss/optimization accurate? AI models are still not good at this unless you write detailed prompts. You might as well do this part on your own. A lot of bad grades came out from over-relying on AI in the past year.
- Presentation: Is your final report publication-ready? Could your final project report be published as a technical report? I don't want to see unprofessional practices that you would not see on a technical paper. Common mistakes: Pixelated images or images not exported as vector graphics first when appropriate as in Figures and plots (automatic points deduction), axis on output figures that don't warrant axes, or axis without labels of what the axis means (automatic points deduction), etc.
- Results: Are your results good? I can judge this by using metrics such as accuracy, BLEU, retrieval@K, human surveys, and I can also judge by looking at any input-output results that you include. I want to SEE you did well.

Overview

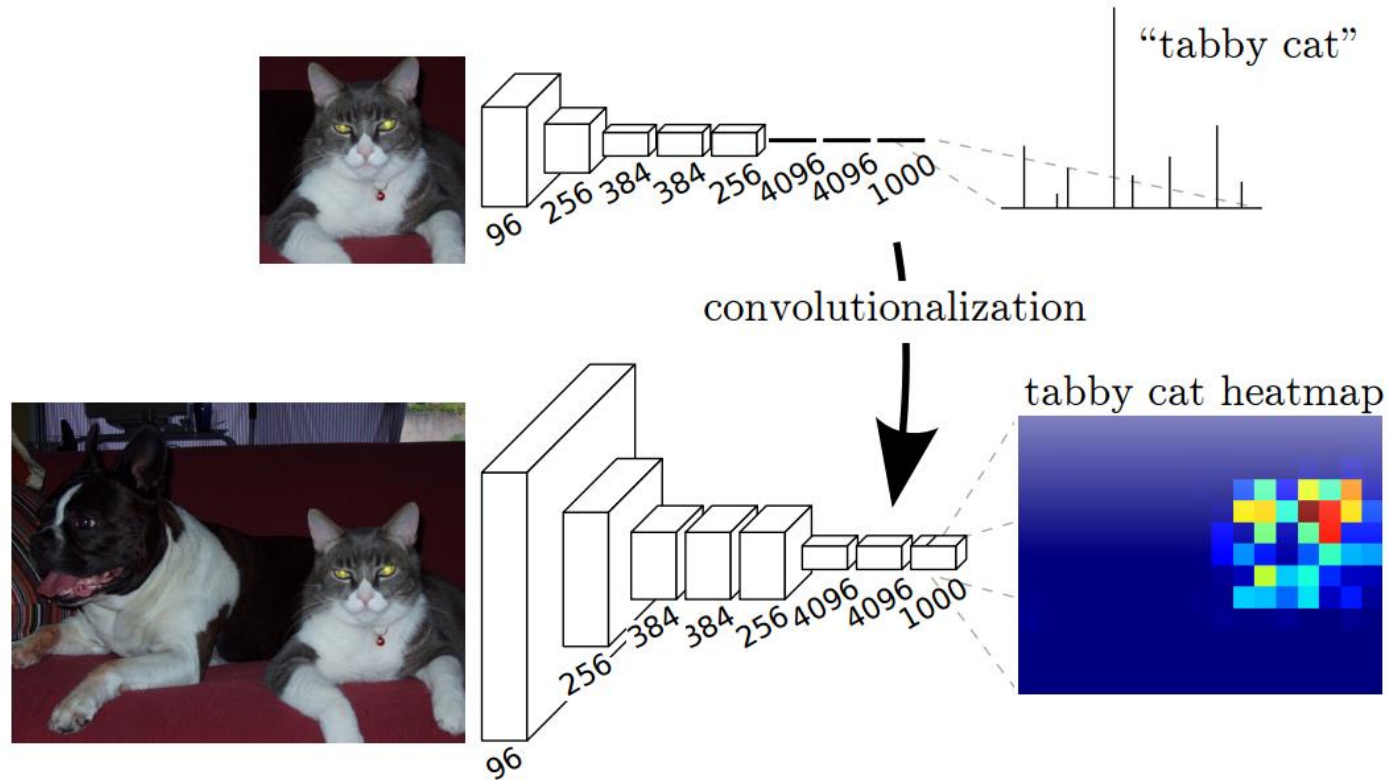
- Image Segmentation
 - Idea 1: Converting Linear layers to Conv layers
 - Idea 2: ConvTranspose2d
 - Idea 3: Dilated Convolutions
- Introduction to Generative Adversarial Networks (GANs)
- Introduction to AutoEncoders (including VAEs)

Semantic Segmentation / Image Parsing



deer
cat
trees
grass

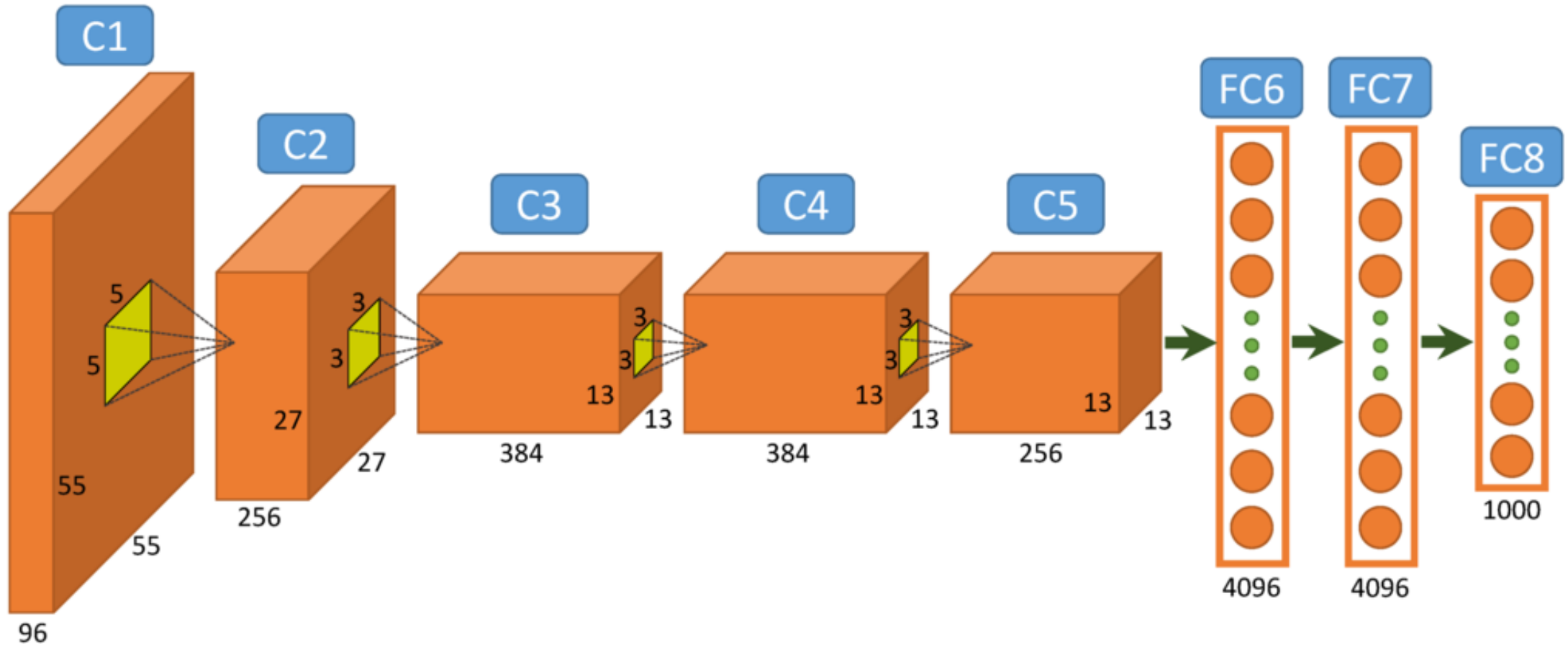
Idea 1: Convolutionalization



However resolution of the segmentation map is low.

https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

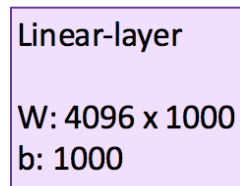
Alexnet



Idea 1: Convolutionalization

`nn.Linear(4096, 1000)` == `nn.Conv2D(4096, 1000, kernel_size = 1, stride = 1)`

input tensor:
4096

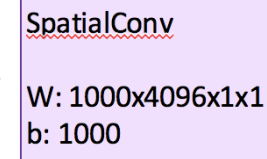


output tensor:
1000



≡

input tensor:
4096x1x1



output tensor:
1000x1x1



Fully Convolutional Networks (CVPR 2015)

Fully Convolutional Networks for Semantic Segmentation

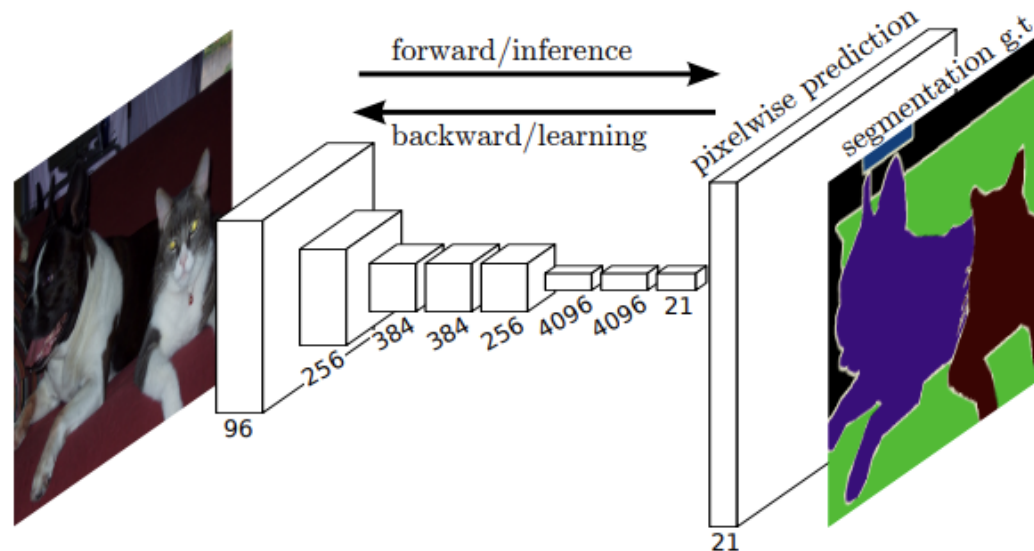
Jonathan Long*

Evan Shelhamer*

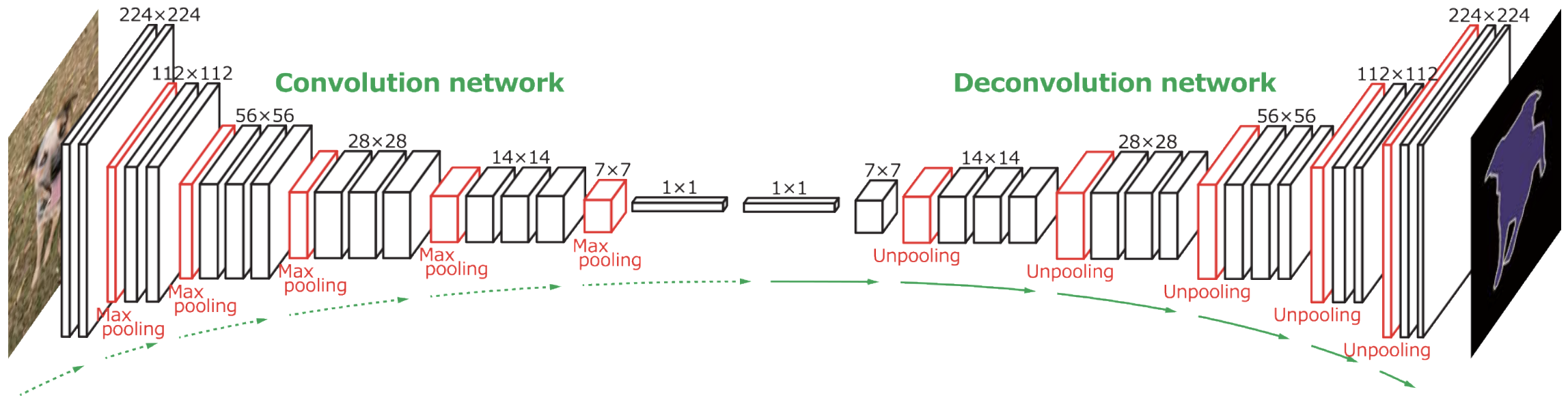
Trevor Darrell

UC Berkeley

{jonlong, shelhamer, trevor}@cs.berkeley.edu



Idea 2: Up-sampling Convolutions or "Deconvolutions" or Transposed Convolutions

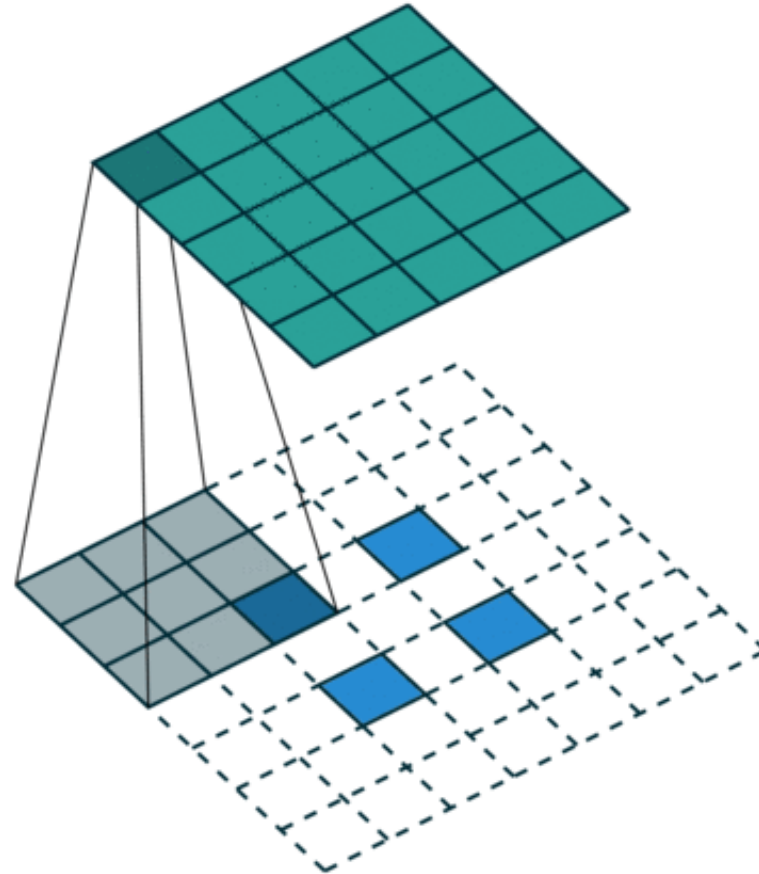


Learning Deconvolution Network for Semantic Segmentation

Hyeonwoo Noh Seunghoon Hong Bohyung Han
Department of Computer Science and Engineering, POSTECH, Korea
{hyeonwoonoh., maga33, bhhan}@postech.ac.kr

<http://cvlab.postech.ac.kr/research/deconvnet/>

Idea 2: Up-sampling Convolutions or "Deconvolutions" or Transposed Convolutions



https://github.com/vdumoulin/conv_arithmetic

Idea 2: Up-sampling Convolutions or “Deconvolutions”

Deconvolutional Layers

Upconvolutional Layers

Backwards Strided
Convolutional Layers

Fractionally Strided
Convolutional Layers

Transposed
Convolutional Layers

Spatial Full
Convolutional Layers

Pytorch

Docs > [torch.nn](#) > ConvTranspose2d



CONVTRANSPOSE2D

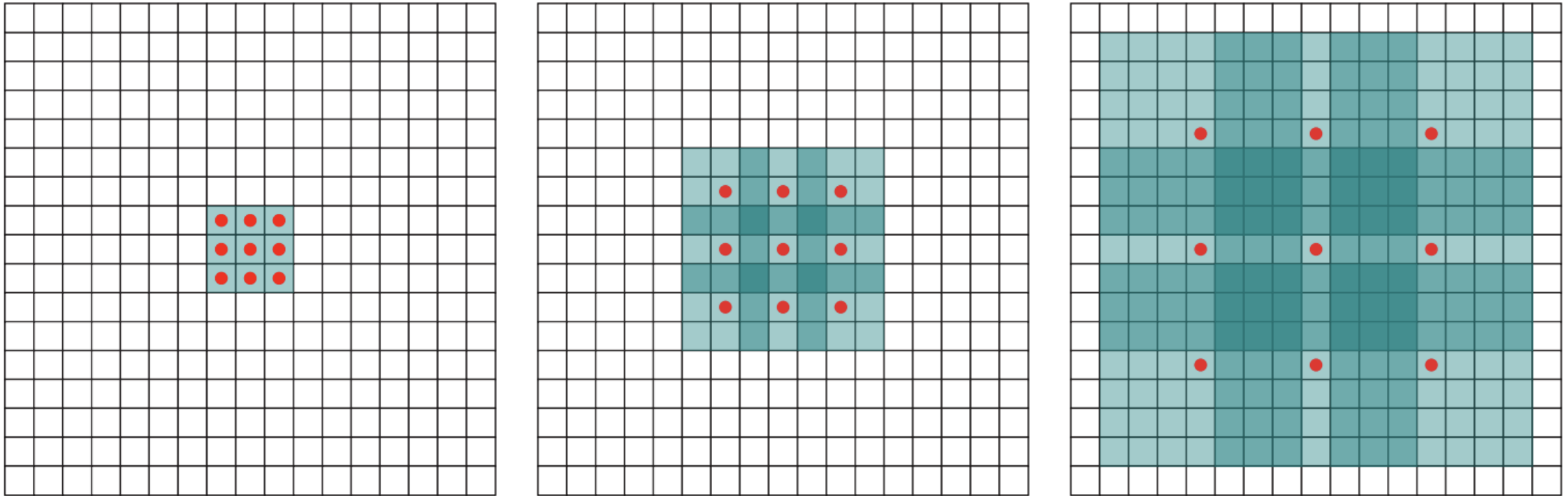
```
CLASS torch.nn.ConvTranspose2d(in_channels, out_channels, kernel_size, stride=1, padding=0,  
    output_padding=0, groups=1, bias=True, dilation=1, padding_mode='zeros', device=None,  
    dtype=None) \[SOURCE\]
```

Applies a 2D transposed convolution operator over an input image composed of several input planes.

This module can be seen as the gradient of Conv2d with respect to its input. It is also known as a fractionally-strided convolution or a deconvolution (although it is not an actual deconvolution operation as it does not compute a true inverse of convolution). For more information, see the visualizations [here](#) and the [Deconvolutional Networks](#) paper.

This module supports [TensorFloat32](#).

Idea 3: Dilated Convolutions



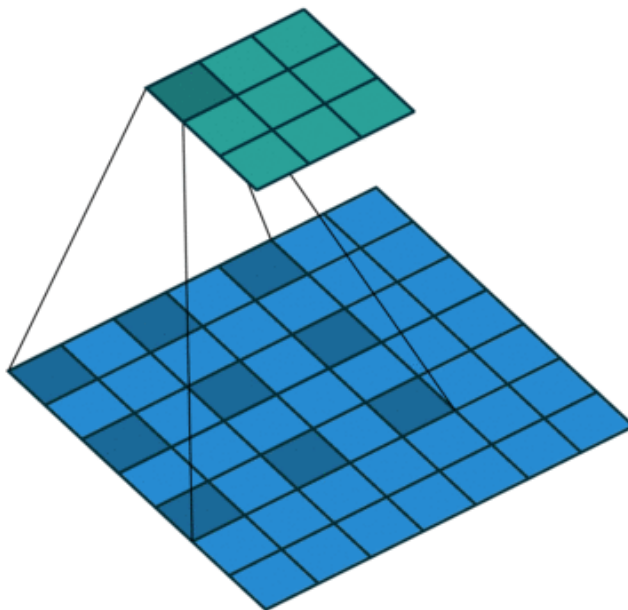
MULTI-SCALE CONTEXT AGGREGATION BY
DILATED CONVOLUTIONS

Fisher Yu
Princeton University

Vladlen Koltun
Intel Labs

ICLR 2016

Idea 3: Dilated Convolutions



MULTI-SCALE CONTEXT AGGREGATION BY
DILATED CONVOLUTIONS

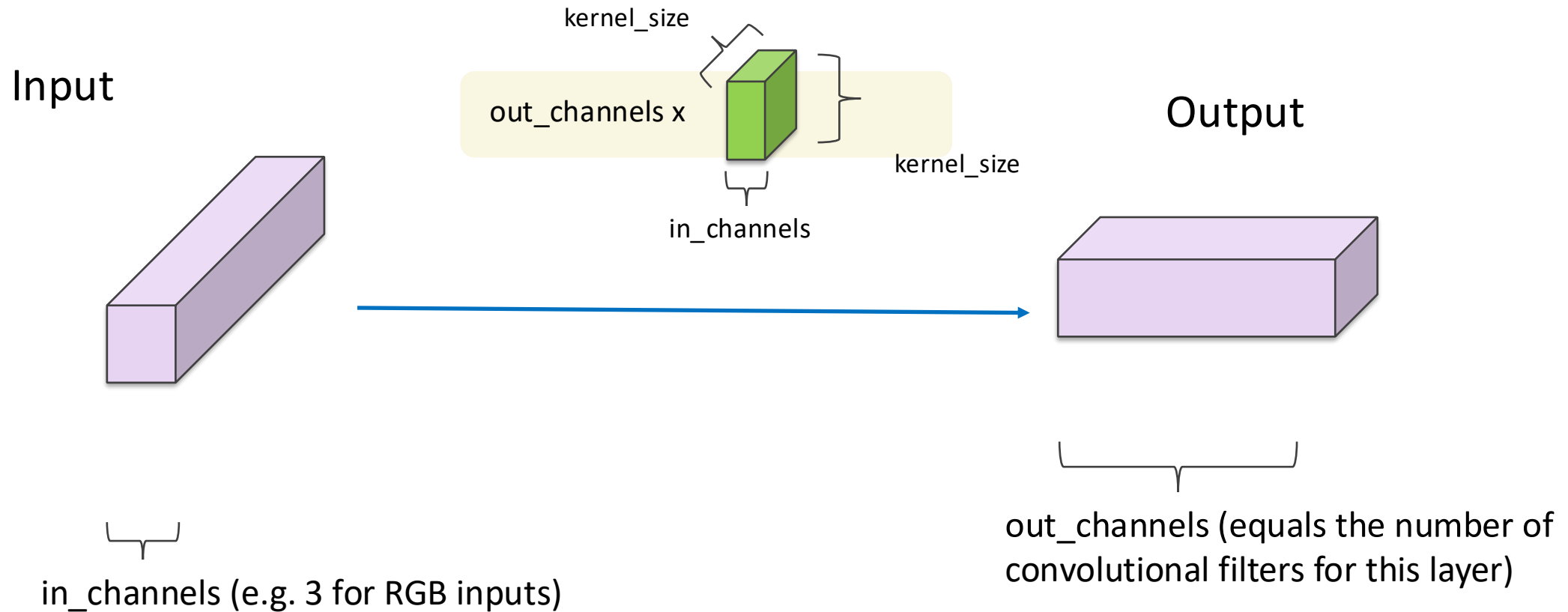
Fisher Yu
Princeton University

Vladlen Koltun
Intel Labs

ICLR 2016

Convolutional Layer in pytorch

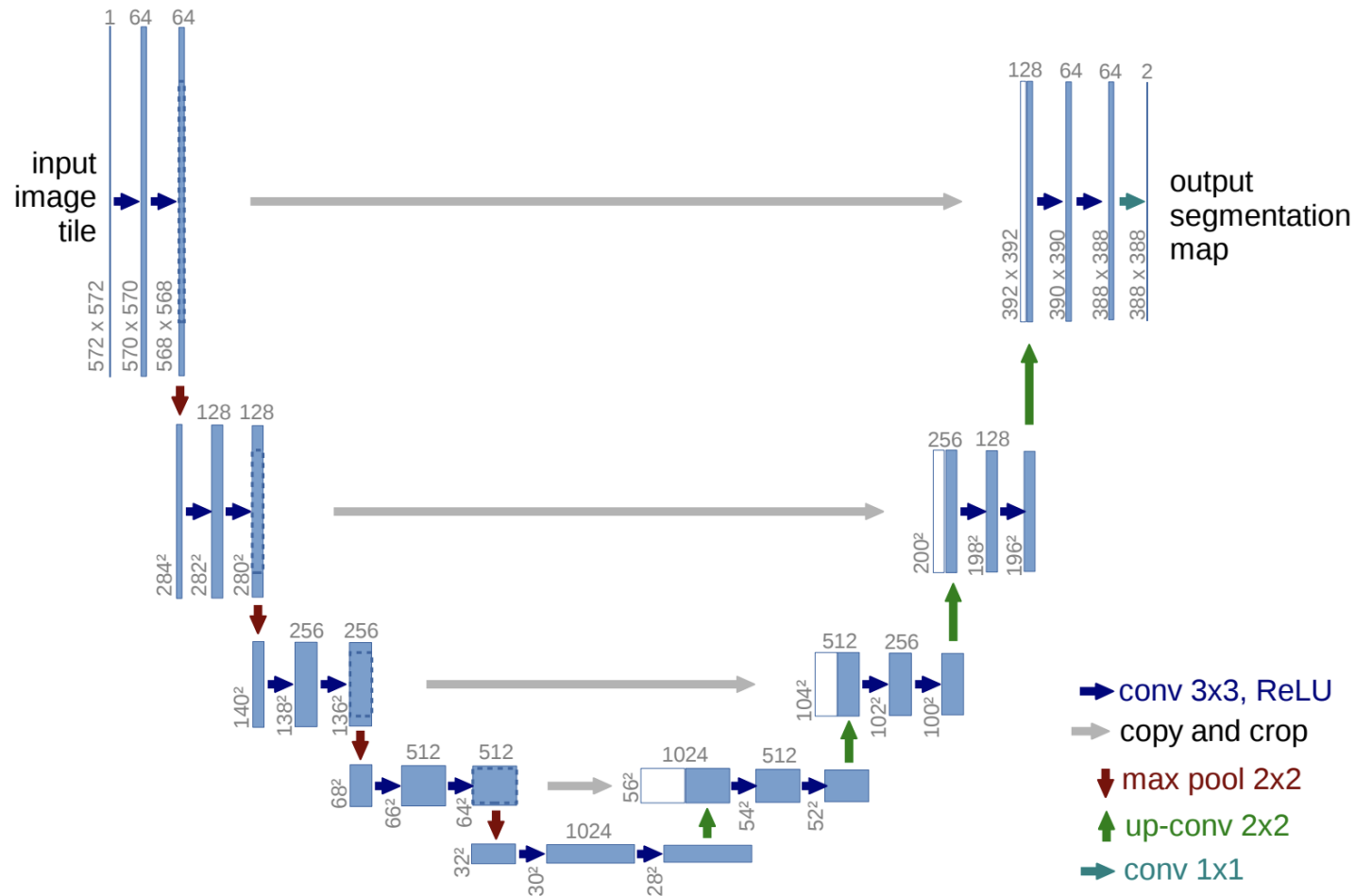
```
class torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1,  
groups=1, bias=True) \[source\]
```



U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOS Centre for Biological Signalling Studies,
University of Freiburg, Germany



<https://arxiv.org/abs/1505.04597>

<https://github.com/milesial/Pytorch-UNet>

<https://github.com/usuyama/pytorch-unet>

UNet in Pytorch

```
from .unet_parts import *

class UNet(nn.Module):
    def __init__(self, n_channels, n_classes, bilinear=False):
        super(UNet, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.bilinear = bilinear

        self.inc = (DoubleConv(n_channels, 64))
        self.down1 = (Down(64, 128))
        self.down2 = (Down(128, 256))
        self.down3 = (Down(256, 512))
        factor = 2 if bilinear else 1
        self.down4 = (Down(512, 1024 // factor))
        self.up1 = (Up(1024, 512 // factor, bilinear))
        self.up2 = (Up(512, 256 // factor, bilinear))
        self.up3 = (Up(256, 128 // factor, bilinear))
        self.up4 = (Up(128, 64, bilinear))
        self.outc = (OutConv(64, n_classes))

    def forward(self, x):
        x1 = self.inc(x)
        x2 = self.down1(x1)
        x3 = self.down2(x2)
        x4 = self.down3(x3)
        x5 = self.down4(x4)
        x = self.up1(x5, x4)
        x = self.up2(x, x3)
        x = self.up3(x, x2)
        x = self.up4(x, x1)
        logits = self.outc(x)
        return logits
```