DEEP LEARNING SEMANTIC SEGMENTATION

Chih-Chung Hsu (許志仲)
Institute of Data Science
National Cheng Kung University
https://cchsu.info

So far: Image Classification

This image is CC0 public domain

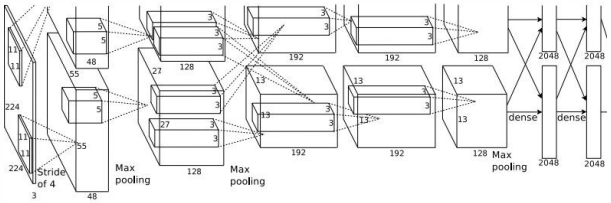


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

Fully-Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

SEMANTIC SEGMENTATION

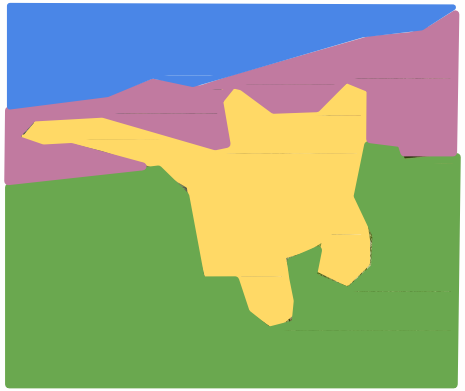
Computer Vision Tasks

Classification

CAT

No spatial extent

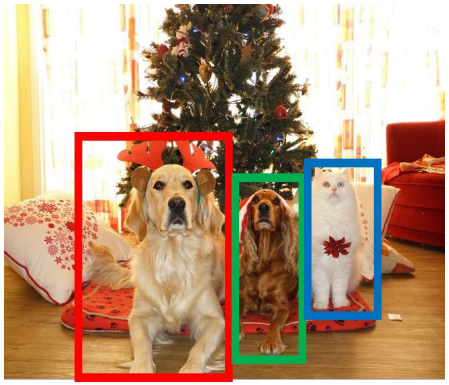
Semantic Segmentation



GRASS, **CAT**, **TREE**, **SKY**

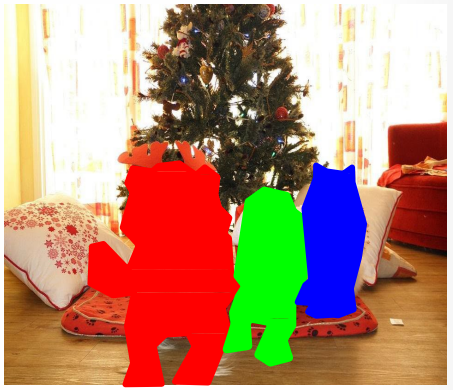
No objects, just pixels

Object Detection



DOG, **DOG**, **CAT**

Instance Segmentation



DOG, **DOG**, **CAT**

Multiple Object

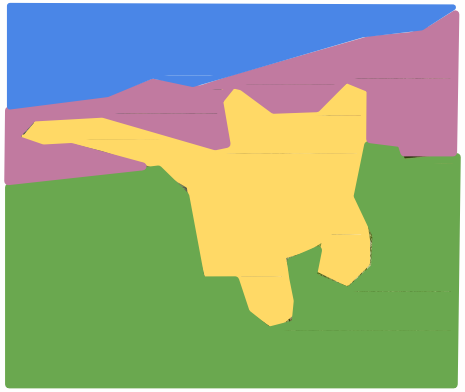
Semantic Segmentation

Classification

CAT

No spatial extent

Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

Multiple Object

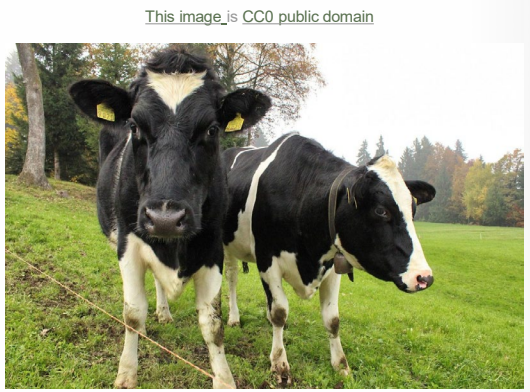
Instance Segmentation

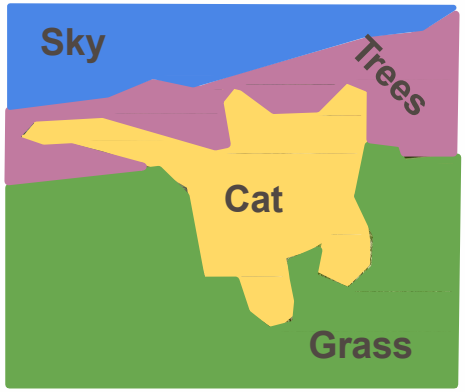
DOG, DOG, CAT

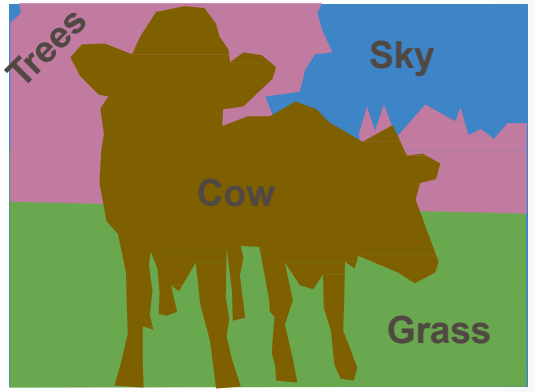
Semantic Segmentation

Label each pixel in the image with a category label

Unnecessary to recognize different instances, only care about pixels







Instance segmentation

- Detect instances, categorize and label every pixel
- Labels are class-aware and instance-aware

Arnab,Torr "Pixelwise instance segmentation with a dynamically instantiated network", CVPR 2017

Datasets for semantic/instance segmentation

Pascal Visual Object Classes

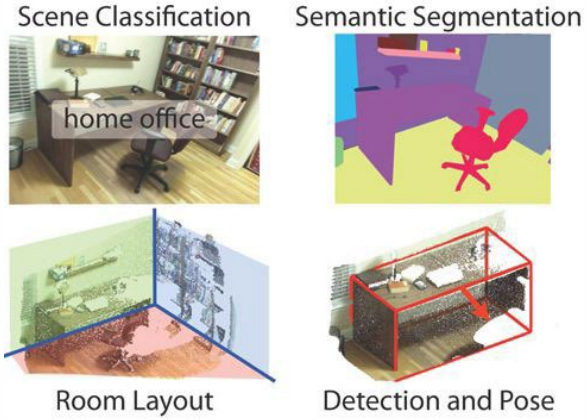
- 20 categories
- +10,000 images
- Semantic segmentation GT
- Instance segmentation GT

Pascal Context

- Real indoor & outdoor scenes
- 540 categories
- +10,000 images
- Dense annotations
- Semantic segmentation GT
- Objects + stuff

Datasets for semantic/instance segmentation

SUN RGB-D



- Real indoor scenes
- 10,000 images
- 58,658 3D bounding boxes
- Dense annotations
- Instances GT
- Semantic segmentation GT
- Objects + stuff

COCO Common Objects in Context

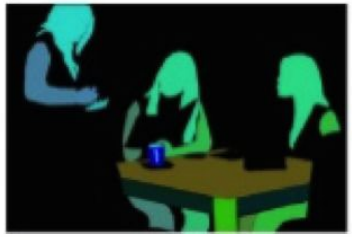
- Real indoor & outdoor scenes
- 80 categories
- +300,000 images
- 2M instances
- Partial annotations
- Semantic segmentation GT
- Instance segmentation GT
- Objects, but no stuff

Datasets for semantic/instance segmentation

CityScapes

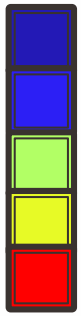
- Real driving scenes
- 30 categories
- +25,000 images
- 20,000 partial annotations
- 5,000 dense annotations
- Semantic segmentation GT
- Instance segmentation GT
- Depth, GPS and other metadata
- Objects and stuff

ADE20K



- Real general scenes
- +150 categories
- +22,000 images
- Semantic segmentation GT
- Instance + parts segmentation GT
- Objects and stuff

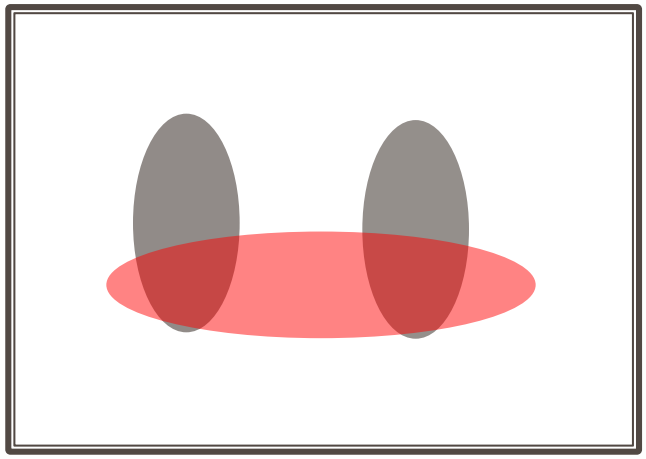
The Task



person
grass
trees
motorbike
road

Evaluation metric

- Pixel classification!
- Accuracy?
  - Heavily unbalanced
- Intersection over Union
  - Average across classes and images
- Per-class accuracy
  - Average across classes and images

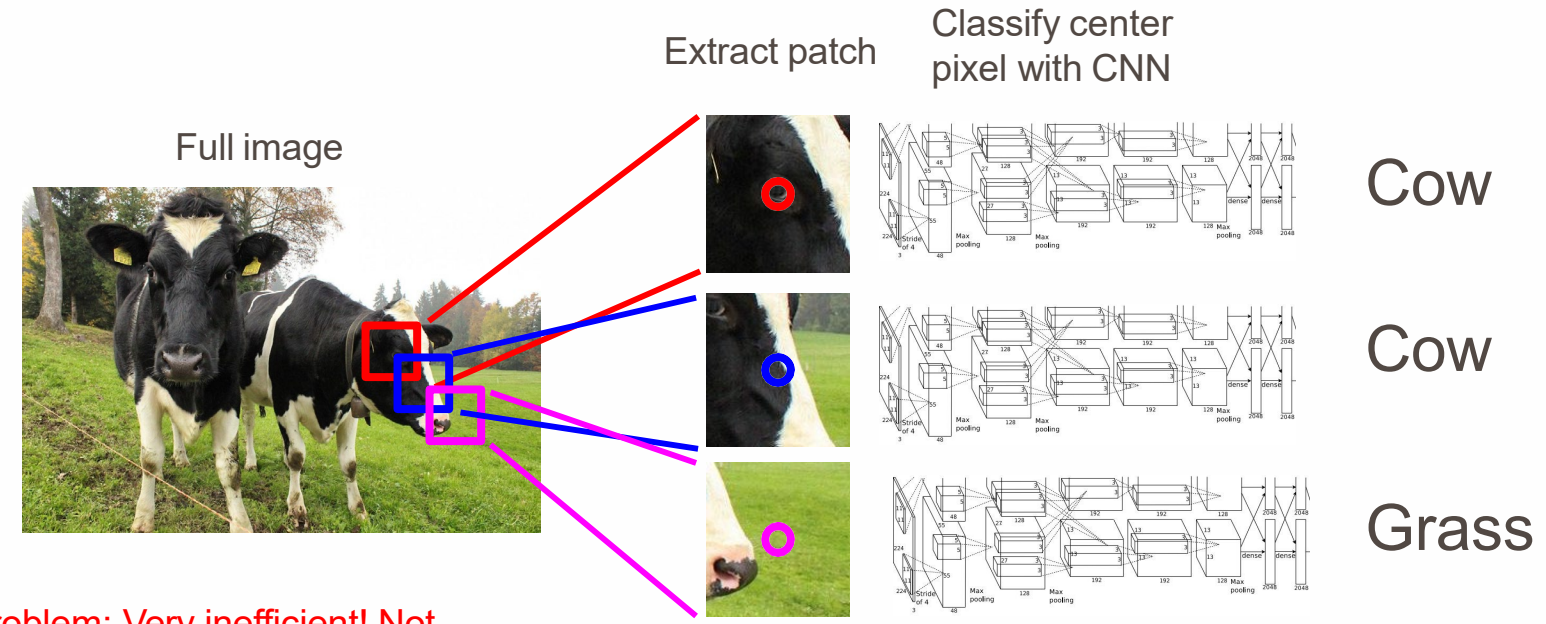


Things vs Stuff

- THINGS
- Person, cat, horse, etc
- Constrained shape
- Individual instances with separate identity
- May need to look at objects

- STUFF
- Road, grass, sky etc
- Amorphous, no shape
- No notion of instances
- Can be done at pixel level
- "texture"

Semantic Segmentation Idea: Sliding Window



Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Challenges in data collection

- Precise localization is hard to annotate
- Annotating every pixel leads to heavy tails
- Common solution: annotate few classes (often things), mark rest as “Other”
- Common datasets: PASCAL VOC 2012 (~1500 images, 20 categories), COCO (~100k images, 20 categories)

Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

Input:
3 x H x W

Conv → Conv → Conv → Conv → argmax

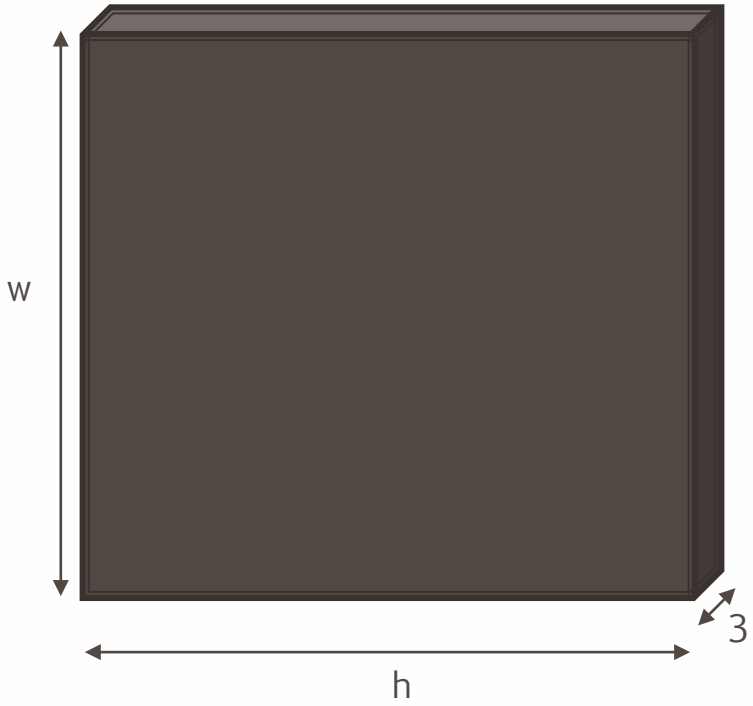
Convolutions:
D x H x W

Scores:
C x H x W

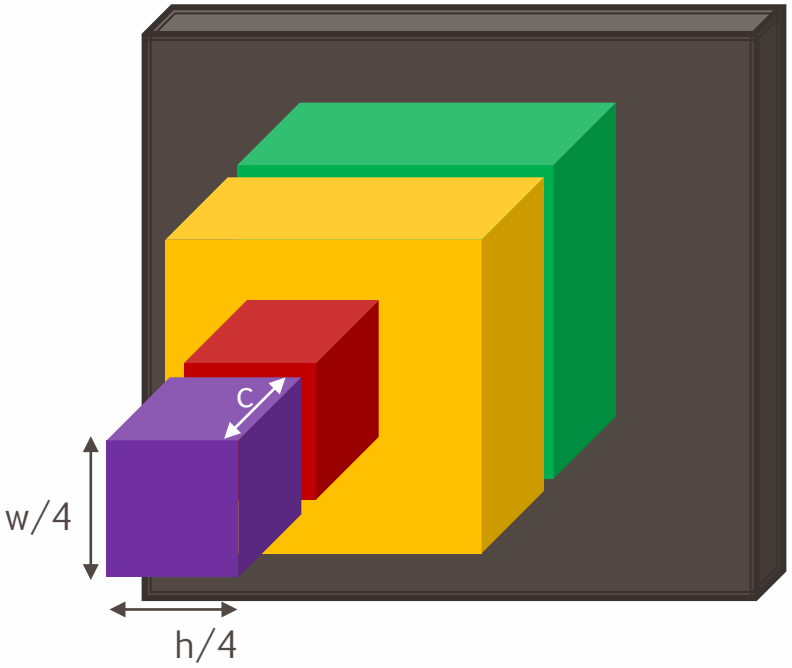
Predictions:
H x W

Problem: convolutions at original image resolution will be very expensive ...

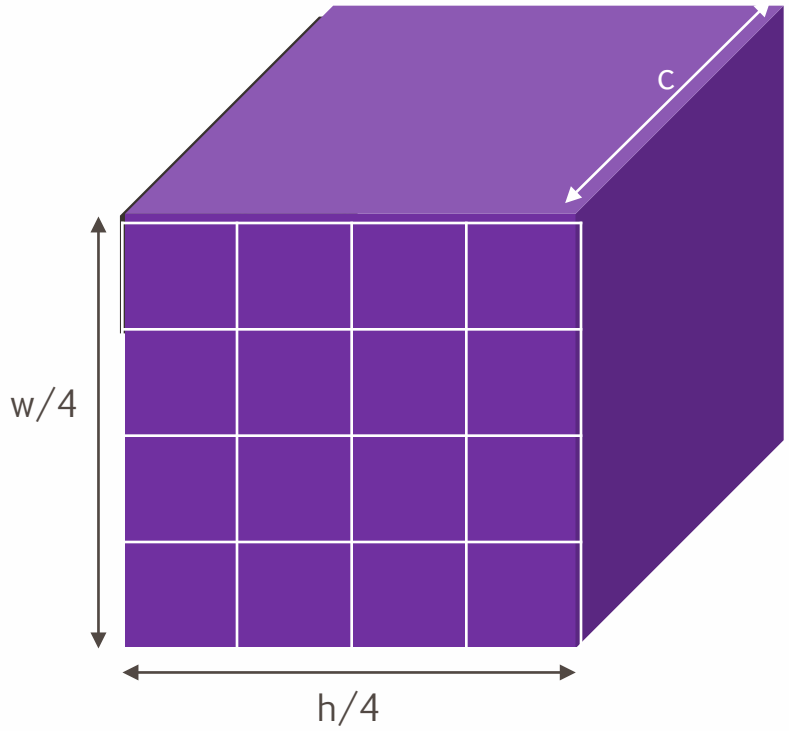
Semantic segmentation using convolutional networks



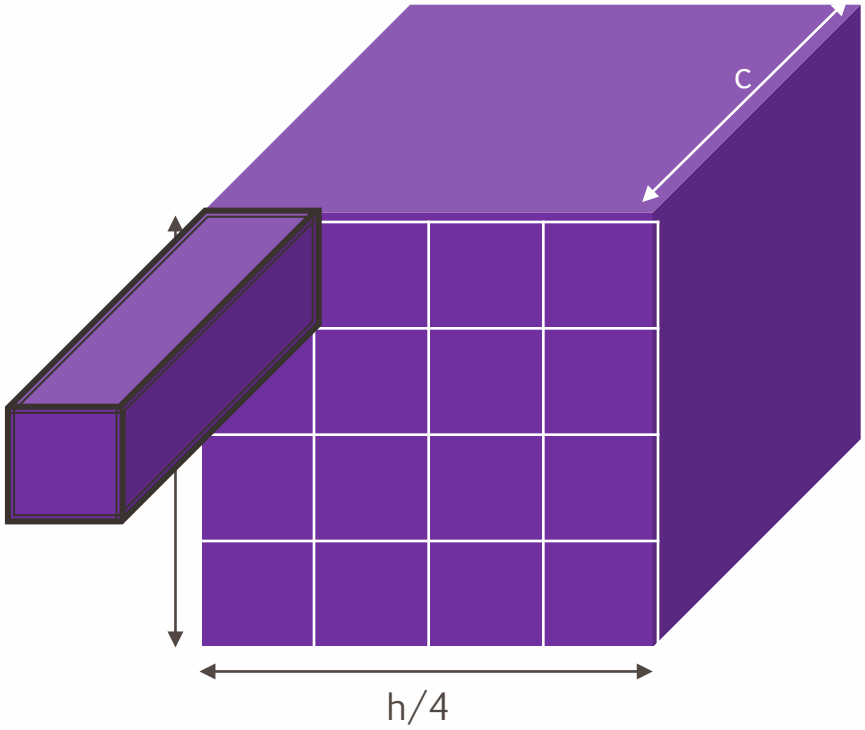
Semantic segmentation using convolutional networks



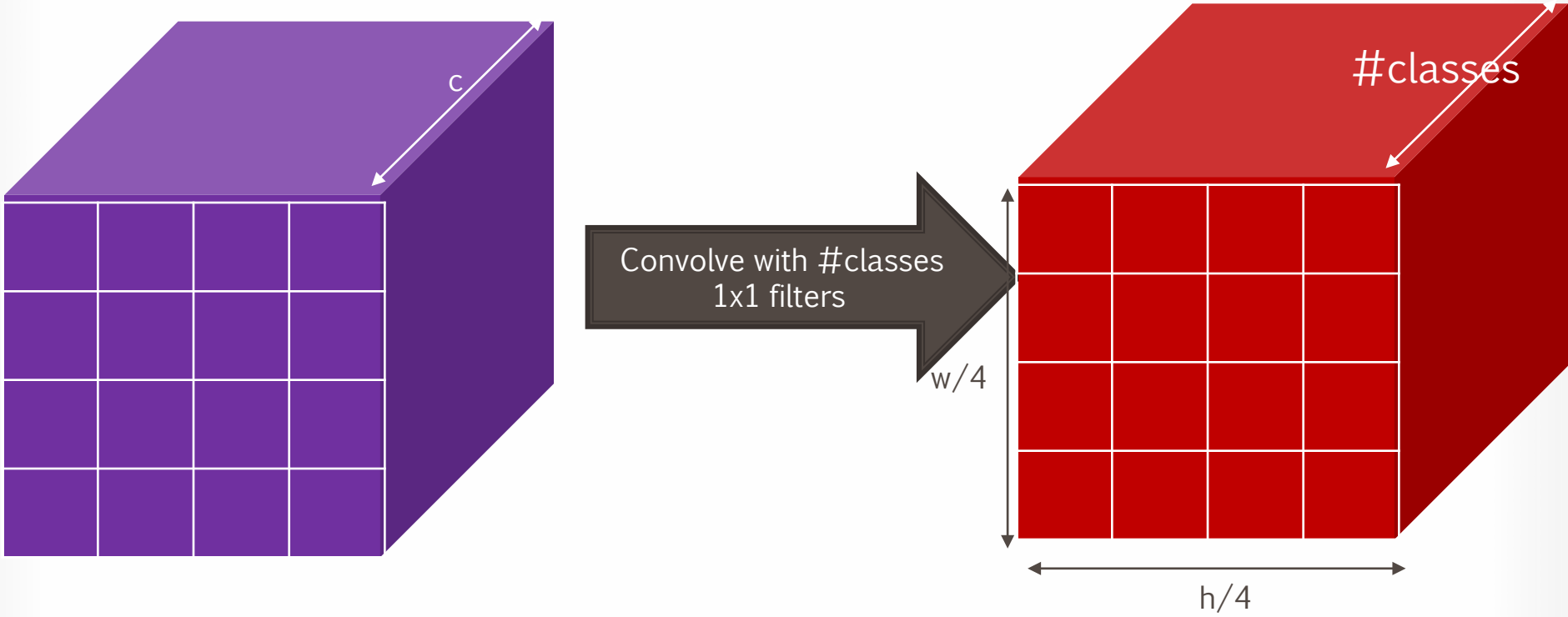
Semantic segmentation using convolutional networks



Semantic segmentation using convolutional networks



Semantic segmentation using convolutional networks



Semantic segmentation using convolutional networks

- Pass image through convolution and subsampling layers
- Final convolution with #classes outputs
- Get scores for subsampled image
- Upsample back to original size

Transfer learning for semantic segmentation

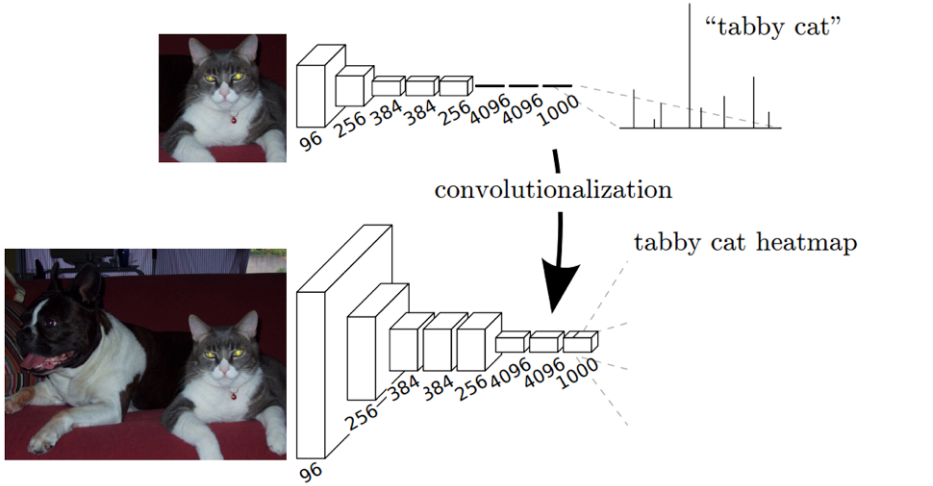
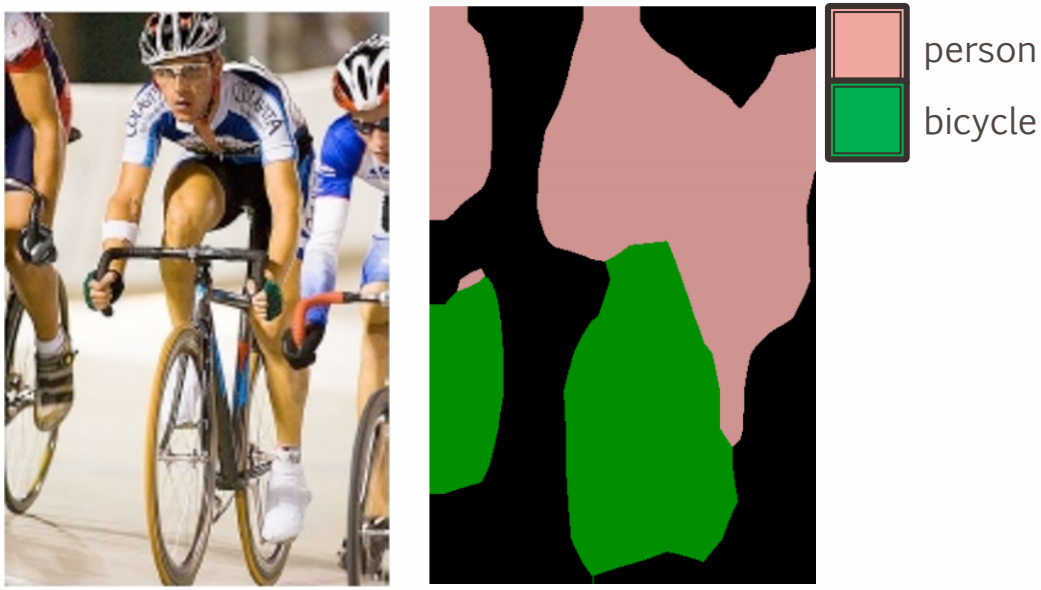


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Semantic segmentation using convolutional networks



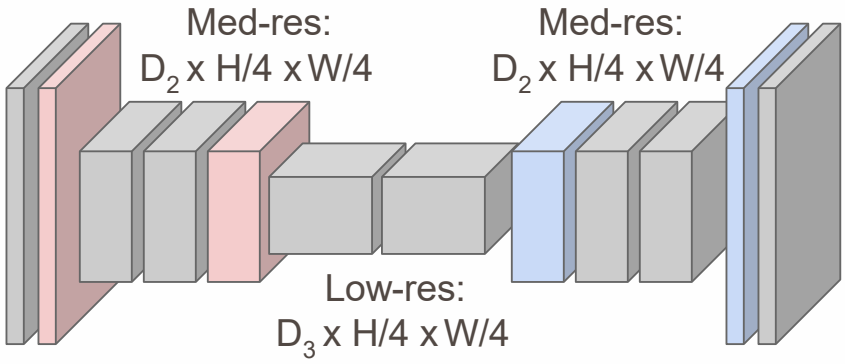
Fully Convolutional Nets

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
???

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

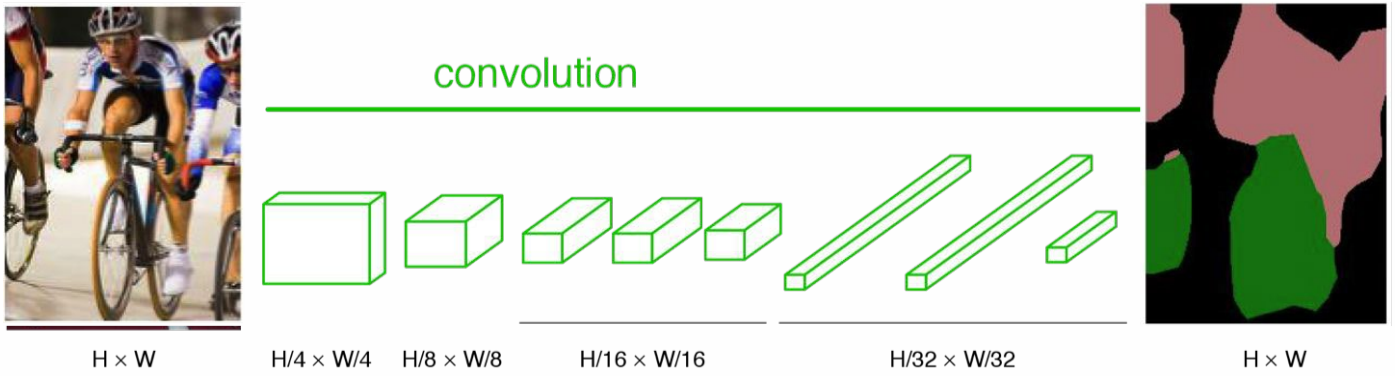
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

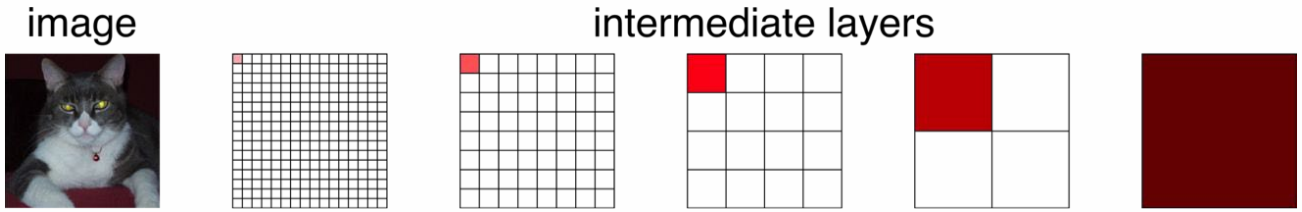
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Resolution: spectrum of deep features

Problem: coarse output



- Combine **where** (local, shallow) with **what** (global, deep)

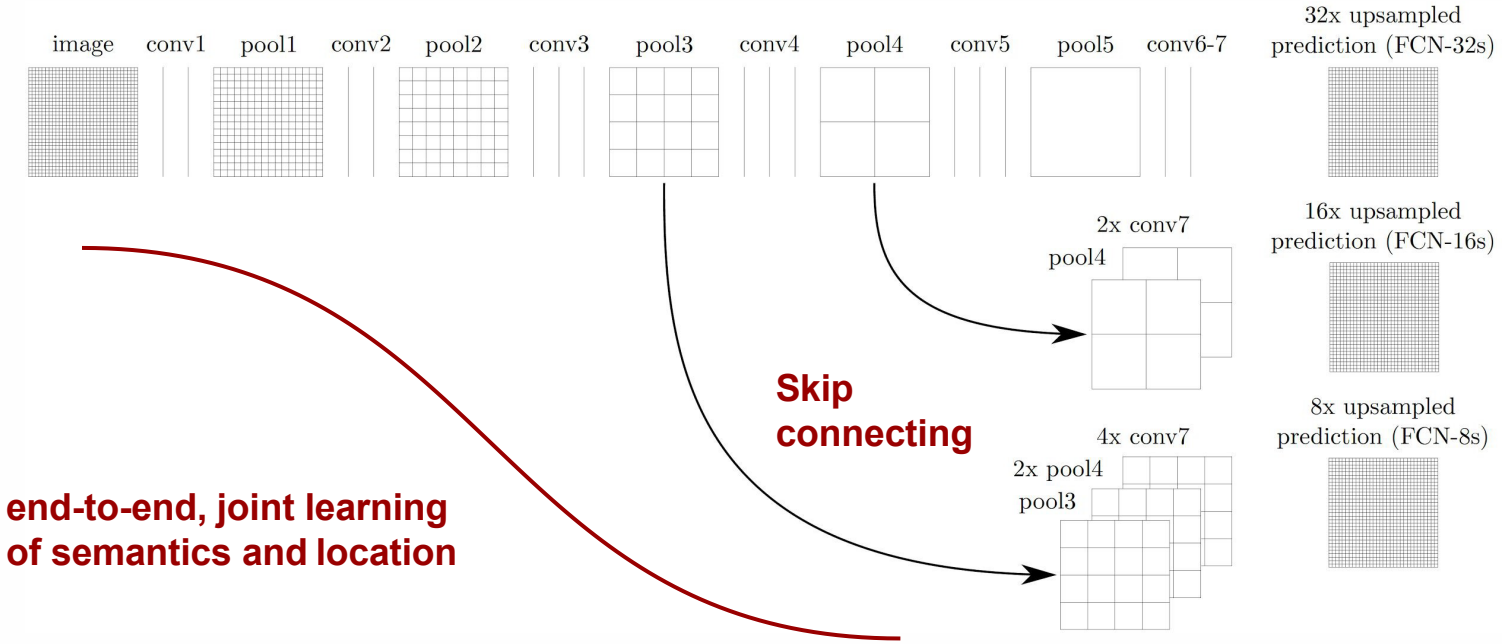


fuse features into deep jet
(cf. Hariharan et al. CVPR15 “hypercolumn”)

Credit: Shelhamer, Long

Fine details: skip connections

Adding 1x1 conv classifying layer on top of pool4, Then upsample x2 (init to bilinear and then learned)
conv7 prediction, sum both, and upsample x16 for output

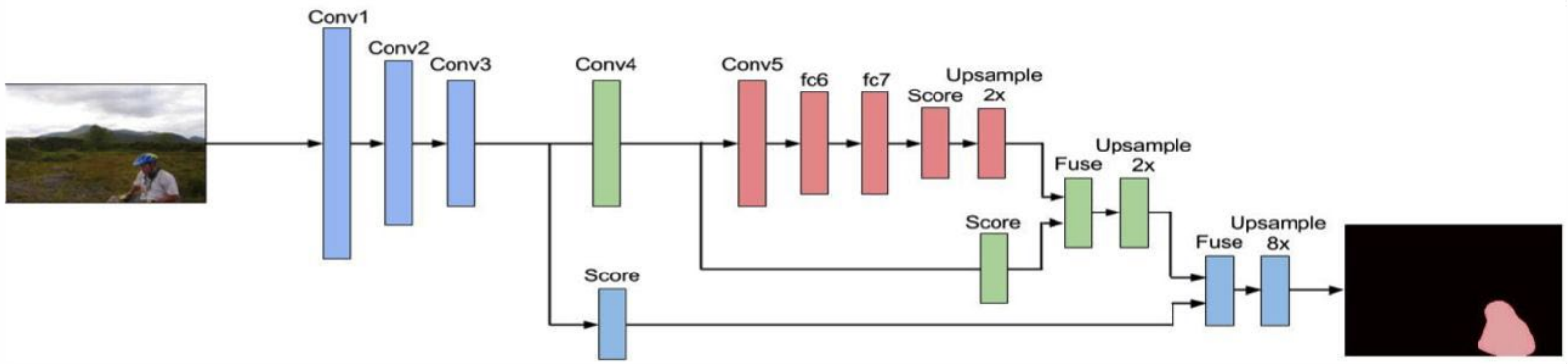


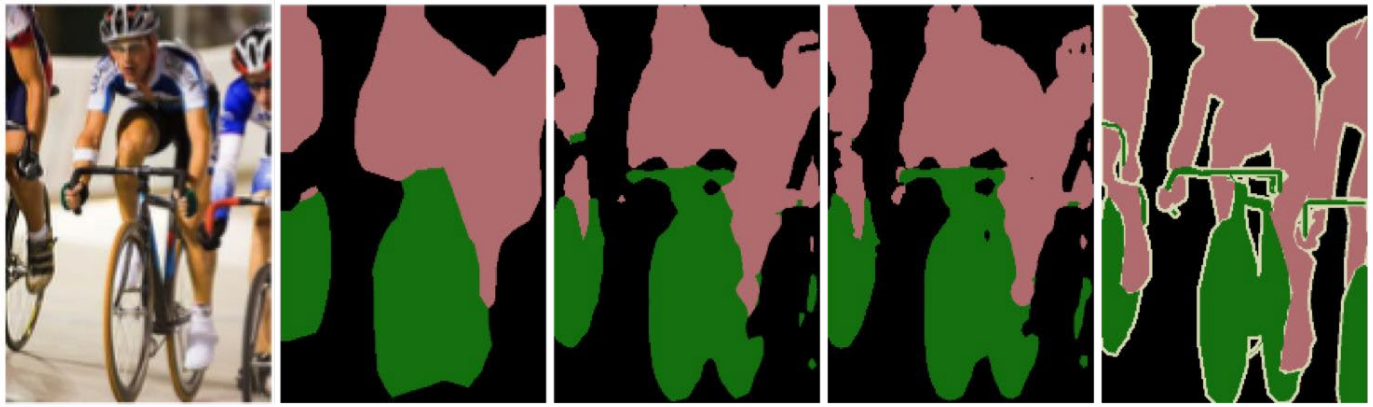
Credit: Shelhamer, Long

19

Skip connections

- A multi-stream network that fuses features/predictions across layers

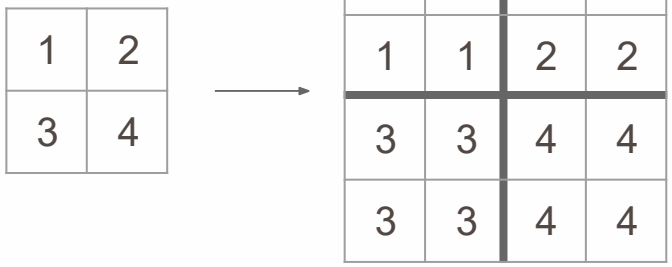




Credit: Shelhamer, Long

In-Network upsampling: "Unpooling"

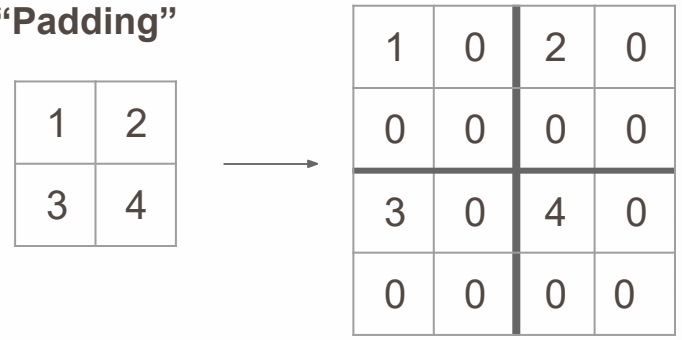
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

"Padding"



Input: 2 x 2

Output: 4 x 4

In-Network upsampling: "Max Unpooling"

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

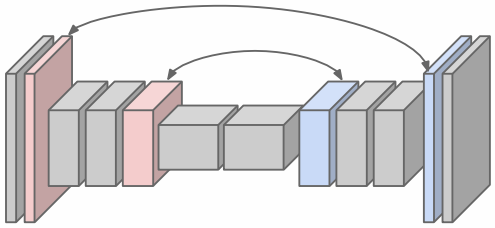
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

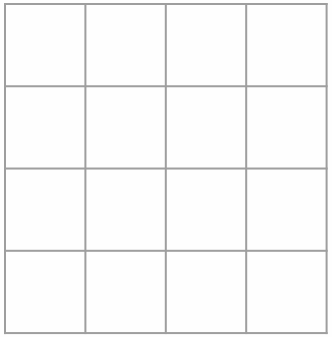
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

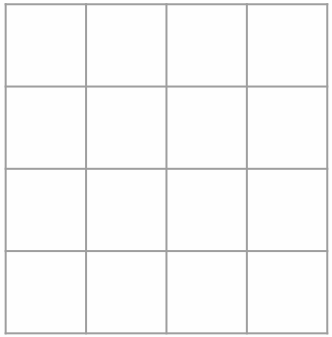


Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



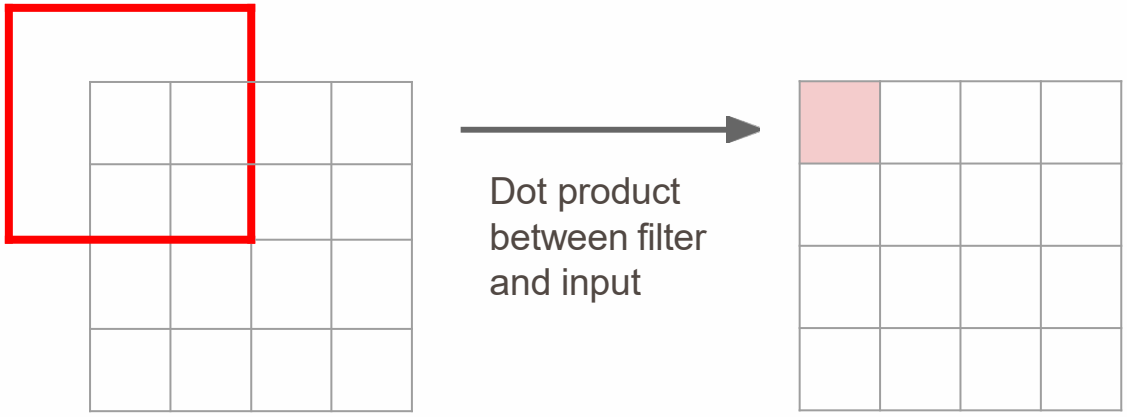
Input: 4 x 4



Output: 4 x 4

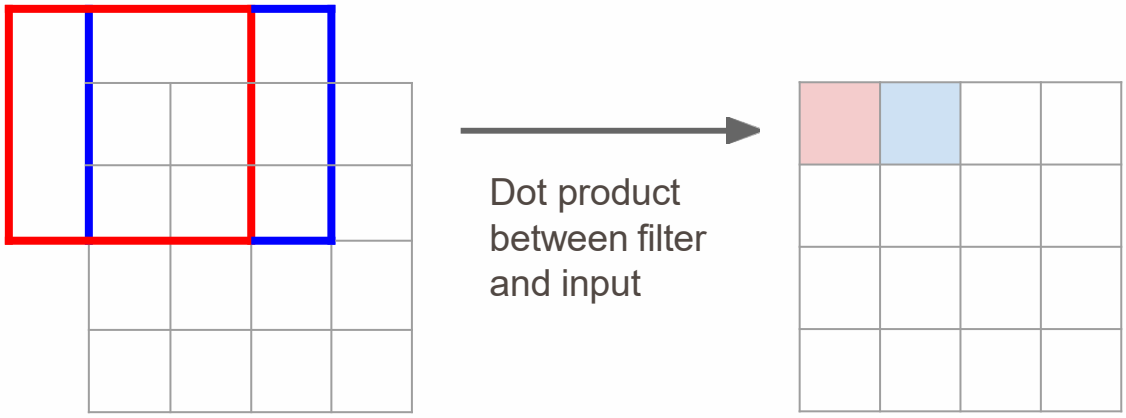
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1

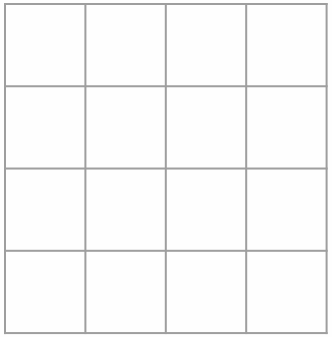


Input: 4 x 4

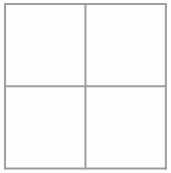
Output: 4 x 4

Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



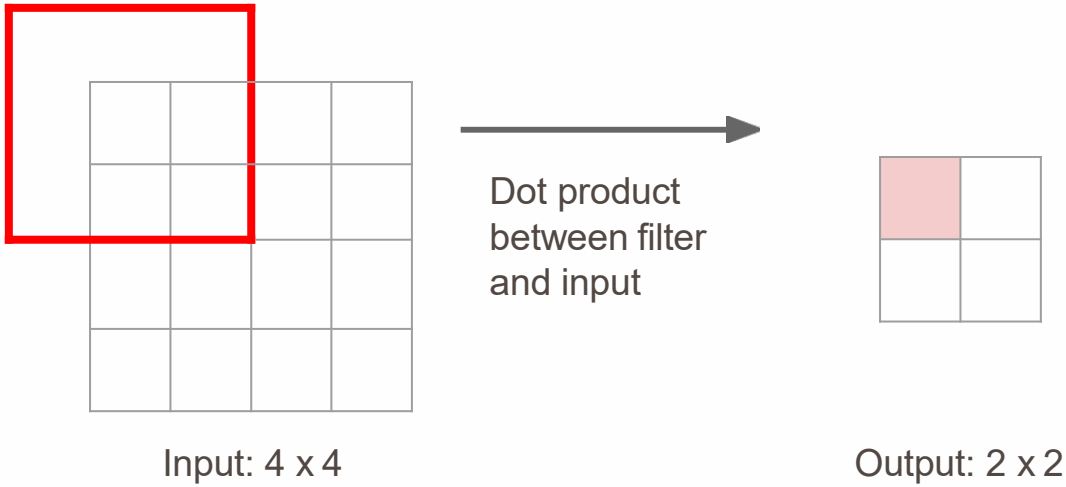
Input: 4 x 4



Output: 2 x 2

Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1

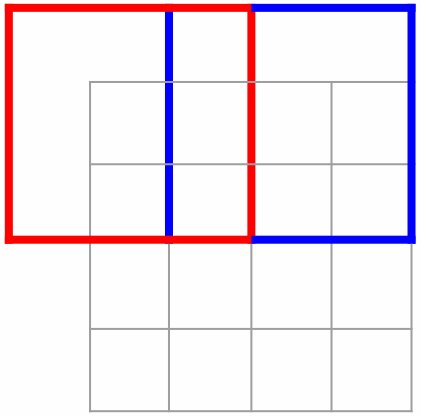


Input: 4 x 4

Output: 2 x 2

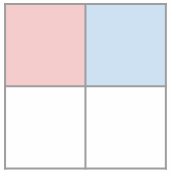
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input



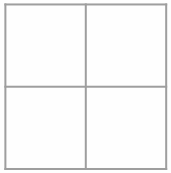
Output: 2 x 2

Filter moves 2 pixels in the input for every one pixel in the output

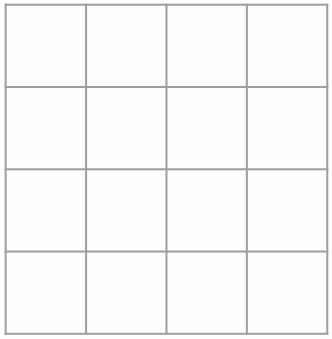
Stride gives ratio between movement in input and output

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



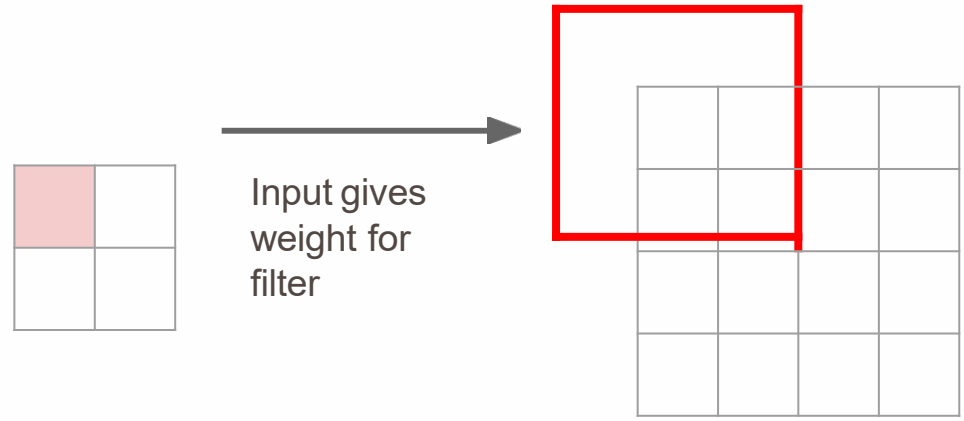
Input: 2 x 2



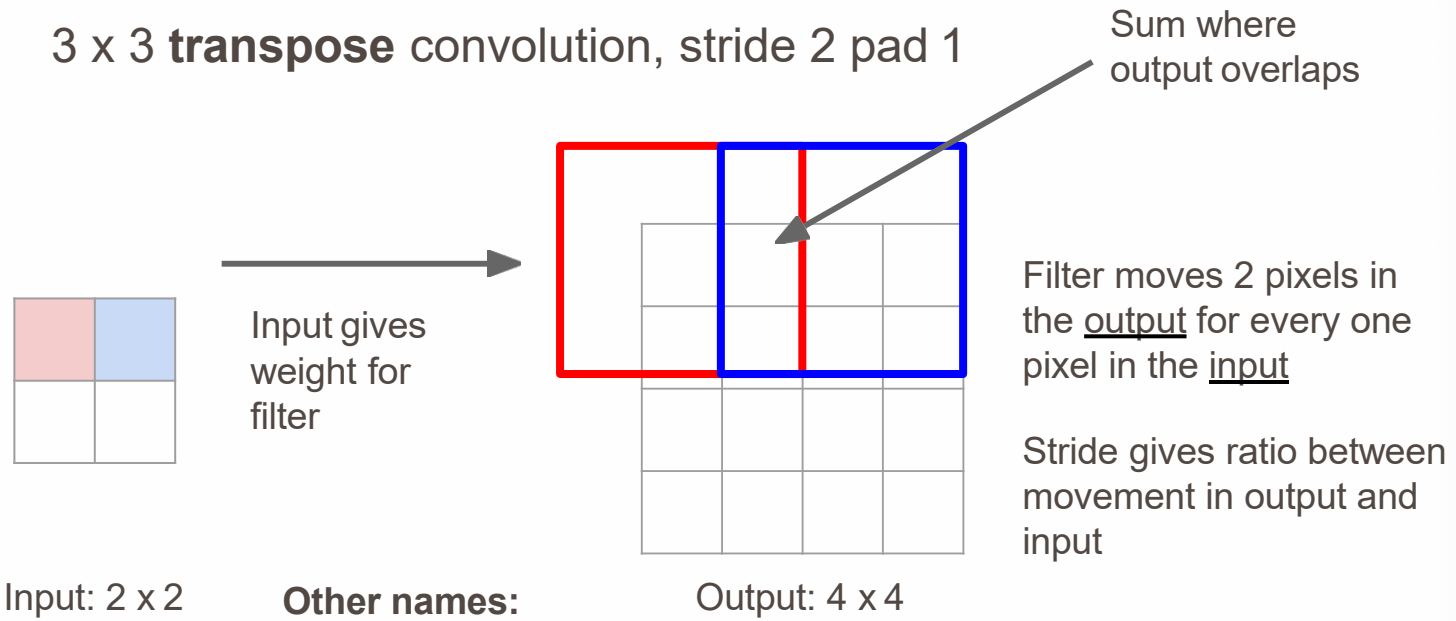
Output: 4 x 4

Learnable Upsampling: Transpose Convolution

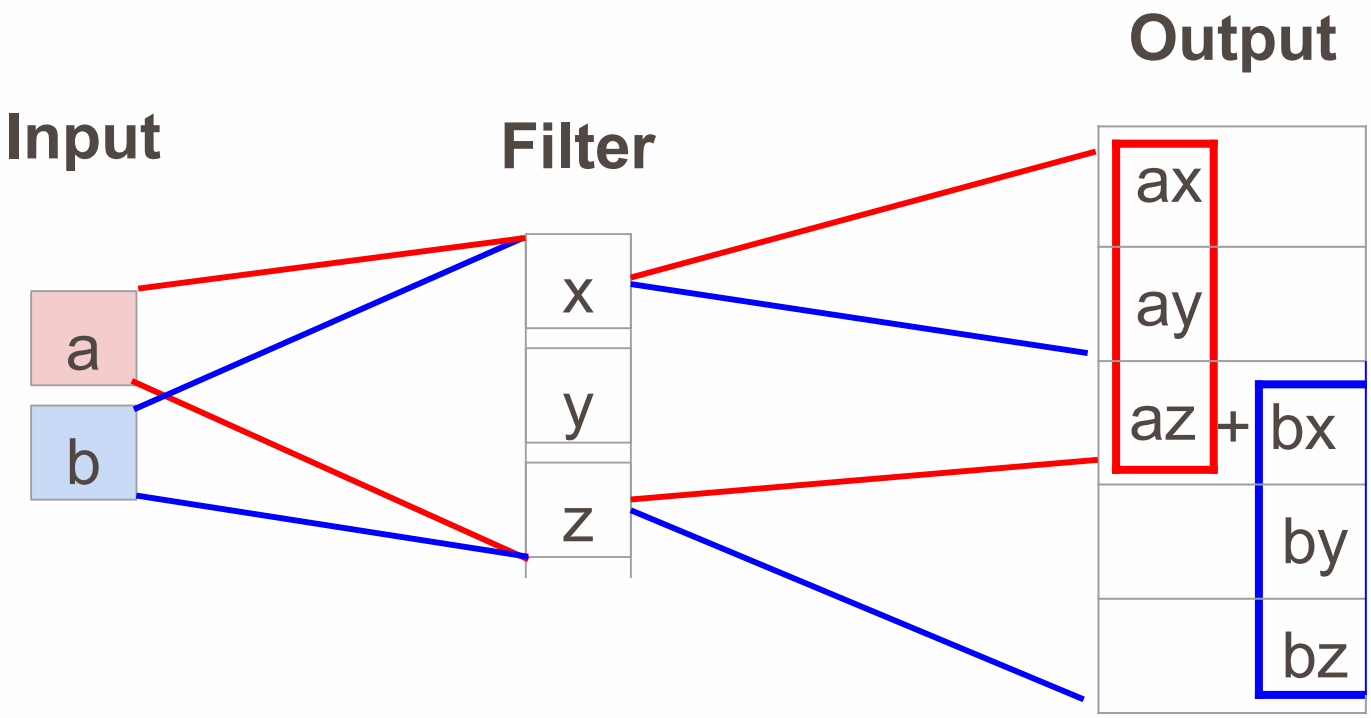
3 x 3 **transpose** convolution, stride 2 pad 1



Learnable Upsampling: Transpose Convolution



Learnable Upsampling: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolutional Kernel

Data

Transposed Convolution multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, transposed convolution tis just a regular convolution (with different padding rules)

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

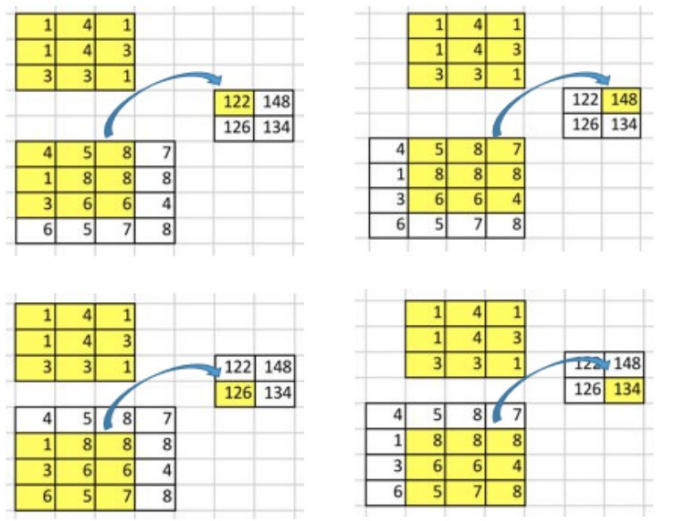
Convolution transpose multiplies by the transpose of the same matrix:

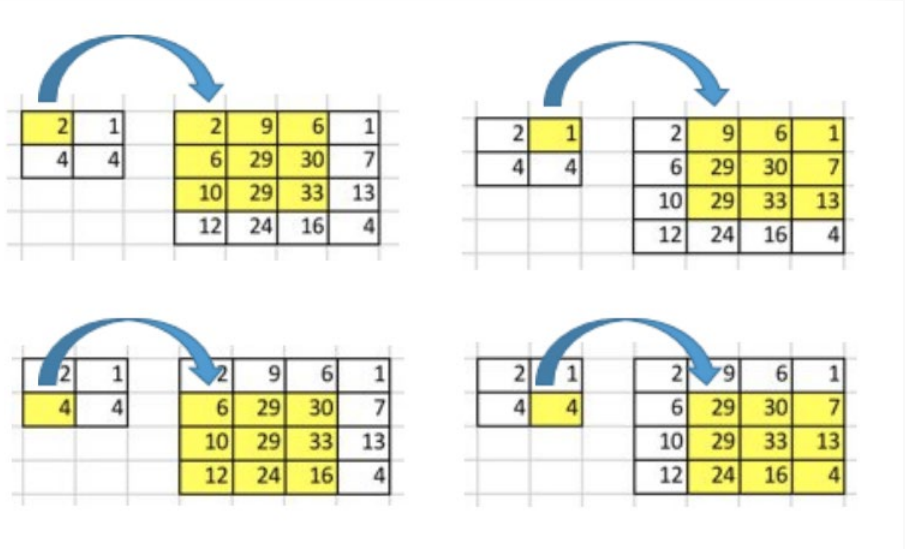
$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

Example: 1D transpose conv, kernel size=3, stride=2, padding=0

2D Conv. vs Transposed Conv.



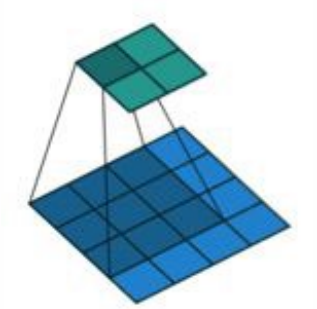


1➔ 9
One result is calculated from the 9 pixels of the input

9 ➔ 1
One pixel should affect 9 pixels of the output as well

Learnable upsampling: recovering spatial shape

Upsampling: Transposed convolution also called fractionally strided convolution or ‘deconvolution’



Convolution

Convolution as a matrix operation

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

$$h=\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}$$

$$O = I * h = C.I$$

I: input image 4x4 vectorized to 16x1
O: output image 4x1 (later reshaped 2x2)
h: 3x3 kernel;
C: 16x4 (weights)

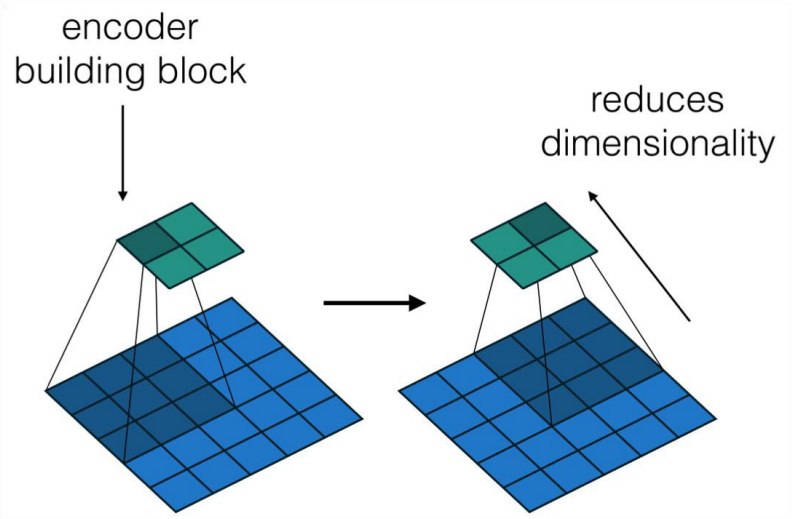
The backward pass is obtained by transposing C: C^T

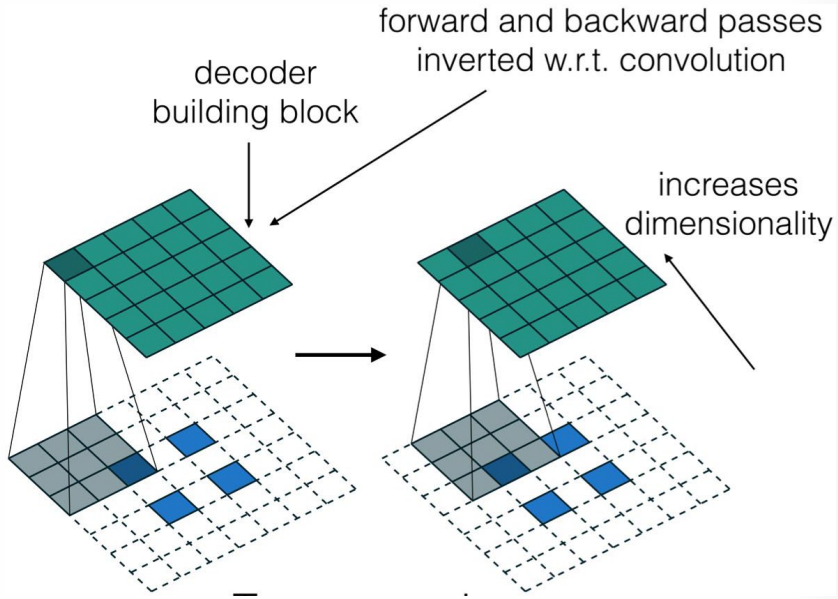
Transposed convolution (also called fractionally strided convolution or ‘deconvolution’): swaps forward and backward passes of a convolution

More info:
Dumoulin et al, A guide to convolution arithmetic for deep learning, 2016 https://github.com/vdumoulin/conv_arithmetic

Learnable upsampling: recovering spatial shape

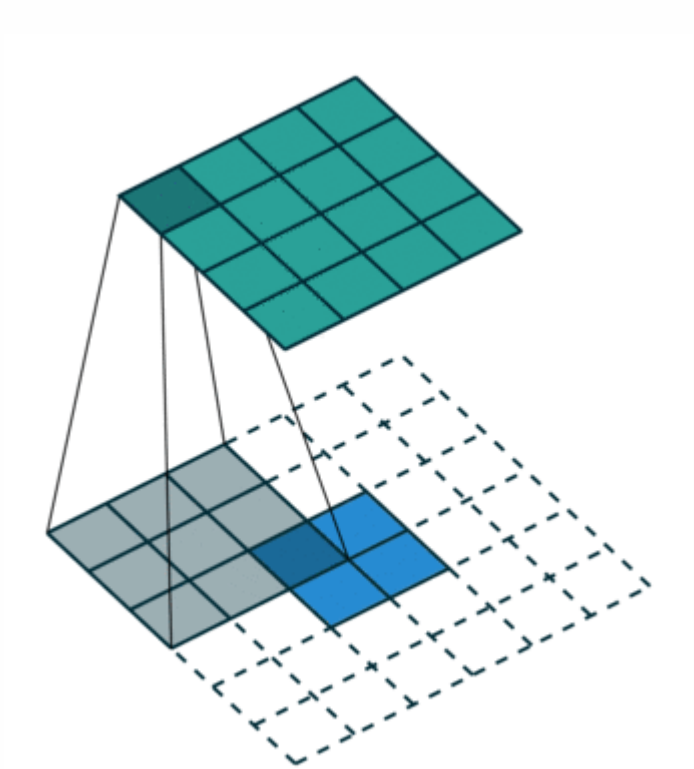
It is always possible to emulate a transposed convolution with a direct convolution (fractional stride)



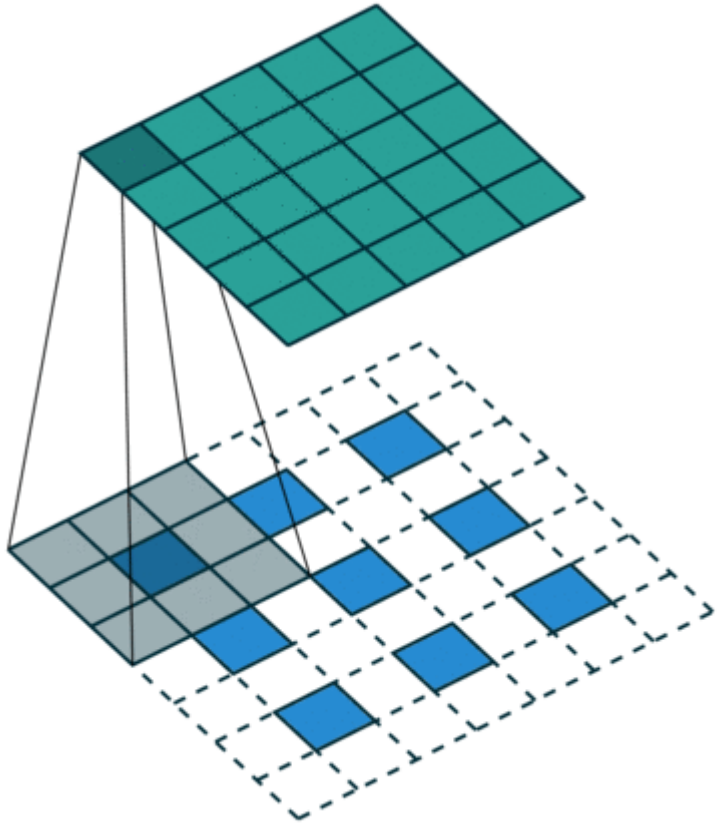


More info:
Dumoulin et al, A guide to convolution arithmetic for deep learning, 2016 https://github.com/vdumoulin/conv_arithmetic

Learnable upsampling: recovering spatial shape



Simple Transposed Conv.



Current Version

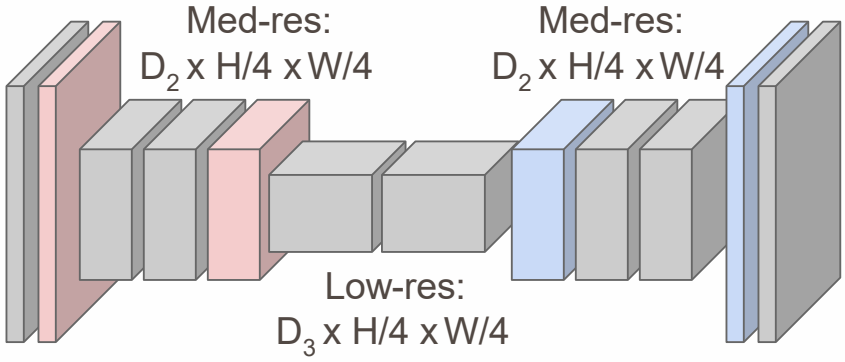
Fully Convolutional Nets

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
Unpooling or strided transpose convolution

Input:
3 x H x W



High-res:
D1 x H/2 x W/2

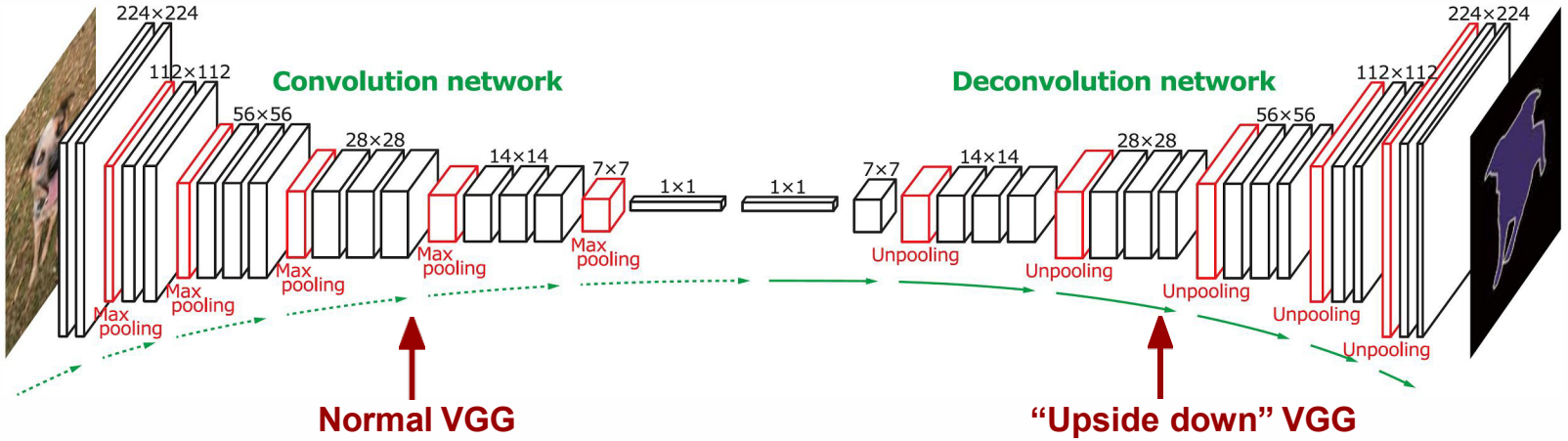
High-res:
D1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

More than one upsampling layer

- DeconvNet:
- VGG-16 (conv+Relu+MaxPool) + mirrored VGG (Unpooling+'deconv'+Relu)



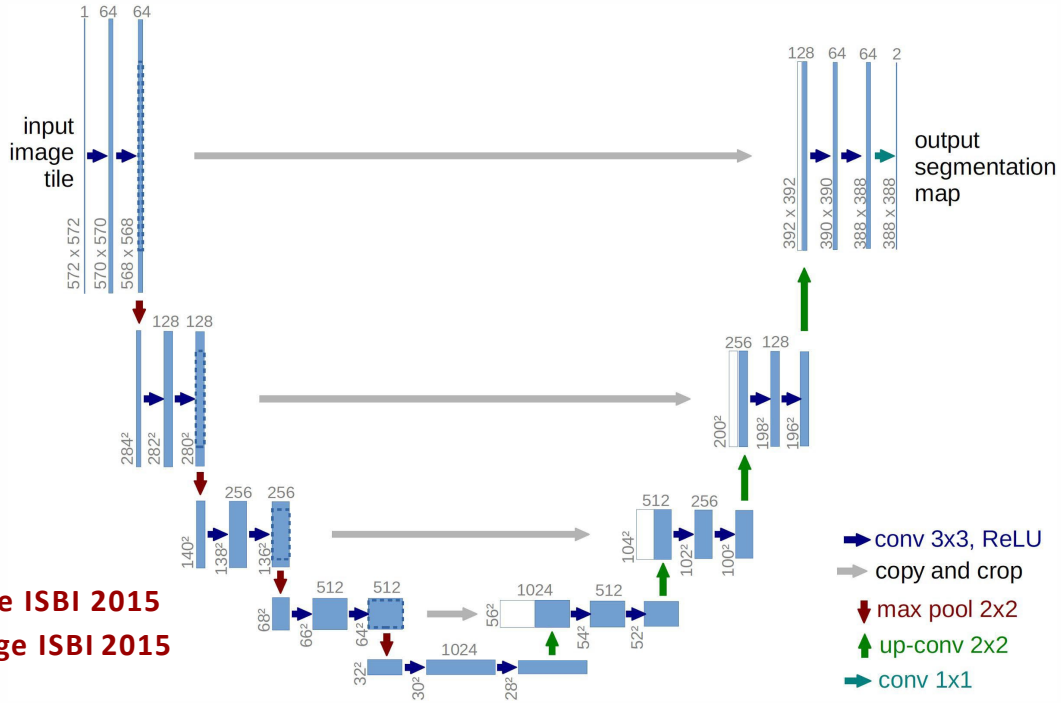
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic segmentation

- Typical architecture
  - Downsampling path: extracts coarse features
  - Upsampling path: recovers input image resolution
  - Skip connections: recovers detailed information
  - Post-processing (optional): refines predictions (CRF)
- Other architectures:
  - DeepLab: ‘atrous’ convolutions + spatial pyramid + CRF (Chen, ICLR 2015)
  - CRF-RNN: FCN + CRF as Recurrent NN (Zheng, ICCV 2015)
  - U-Net (Ronnemberger, 2015)
  - Fully Convolutional DenseNets (Jégou, 2016)
  - Dilated convolutions (Yu, 2016)

U-Net

- A contracting path and an expansive path
- Adds convolutions in the upsampling path (“symmetric” net)
- Skip connections: concatenation of feature maps



Winner of
CAD Caries challenge ISBI 2015
Cell tracking challenge ISBI 2015

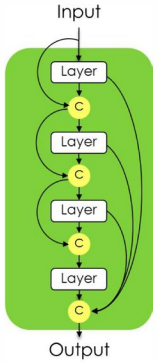
Ronneberger et al, “U-Net: Convolutional Networks for Biomedical Image Segmentation”, arXiv 2015

24

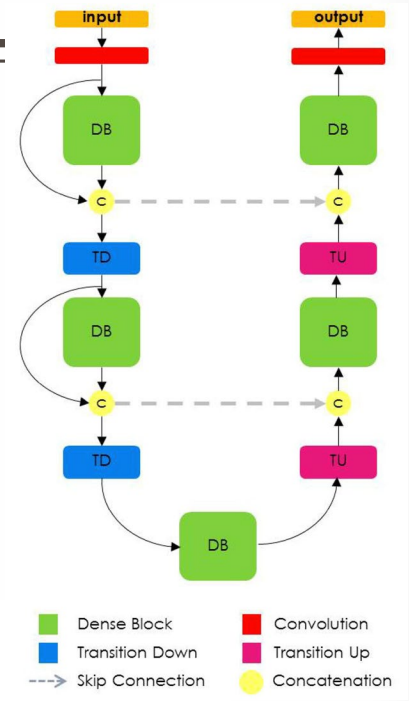
Fully convolutional DenseNets

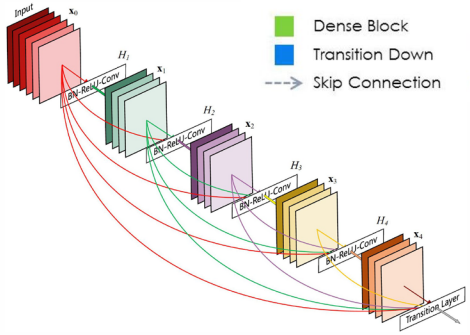
- Adds feed-forward connections between layers

Dense block:



Complete architecture:



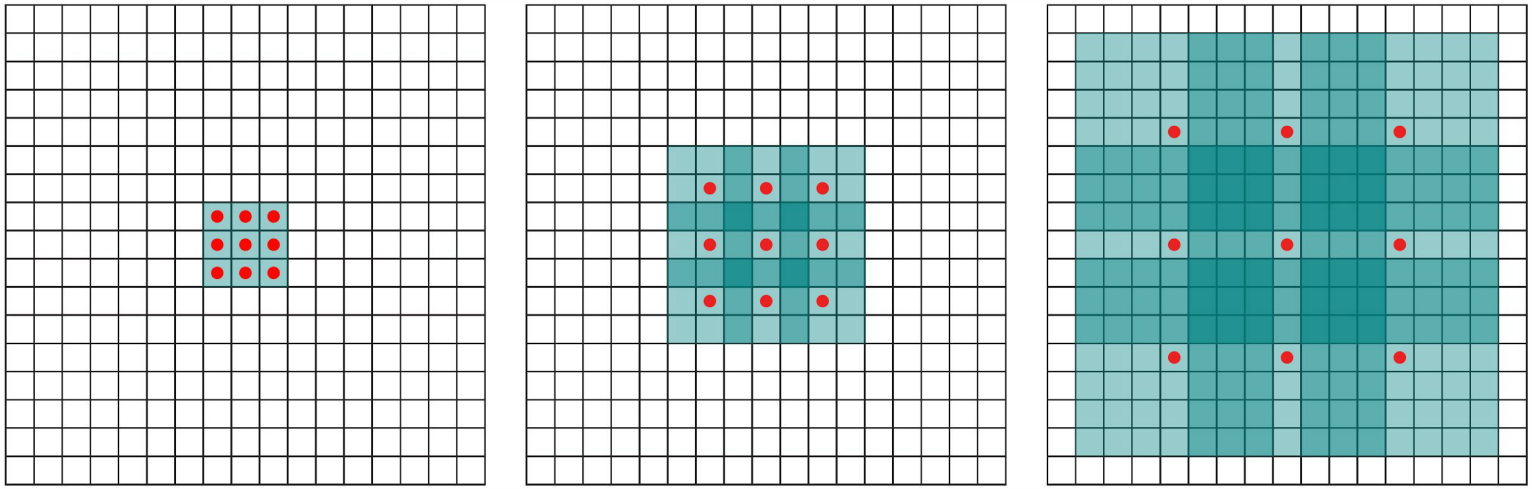


- **Based on U-Nets:**
  - connections between downsampling – upsampling paths
- **Based on DenseNets*** (for image classification):
  - each layer directly connected to every other layer
  - alleviate the vanishing-gradient problem
  - strengthen feature propagation
  - encourage feature reuse
  - substantially reduce the number of parameters.

Jégou et al, "The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation", Dec. 2016
*Huang et al, "Densely connected convolutional networks", arxiv Aug 2016

Dilated convolutions

- Systematically aggregate multiscale contextual information without losing resolution
  - Usual convolution $(F * k)(\mathbf{p}) = \sum_{\mathbf{s}+\mathbf{t}=\mathbf{p}} F(\mathbf{s})\, k(\mathbf{t})$ $(F * k)(\mathbf{p}) = \sum F(\mathbf{s})\, k(\mathbf{t})$
  - Dilated convolution $(F *_l k)(\mathbf{p}) = \sum_{\mathbf{s}+l\mathbf{t}=\mathbf{p}} F(\mathbf{s})\, k(\mathbf{t})$

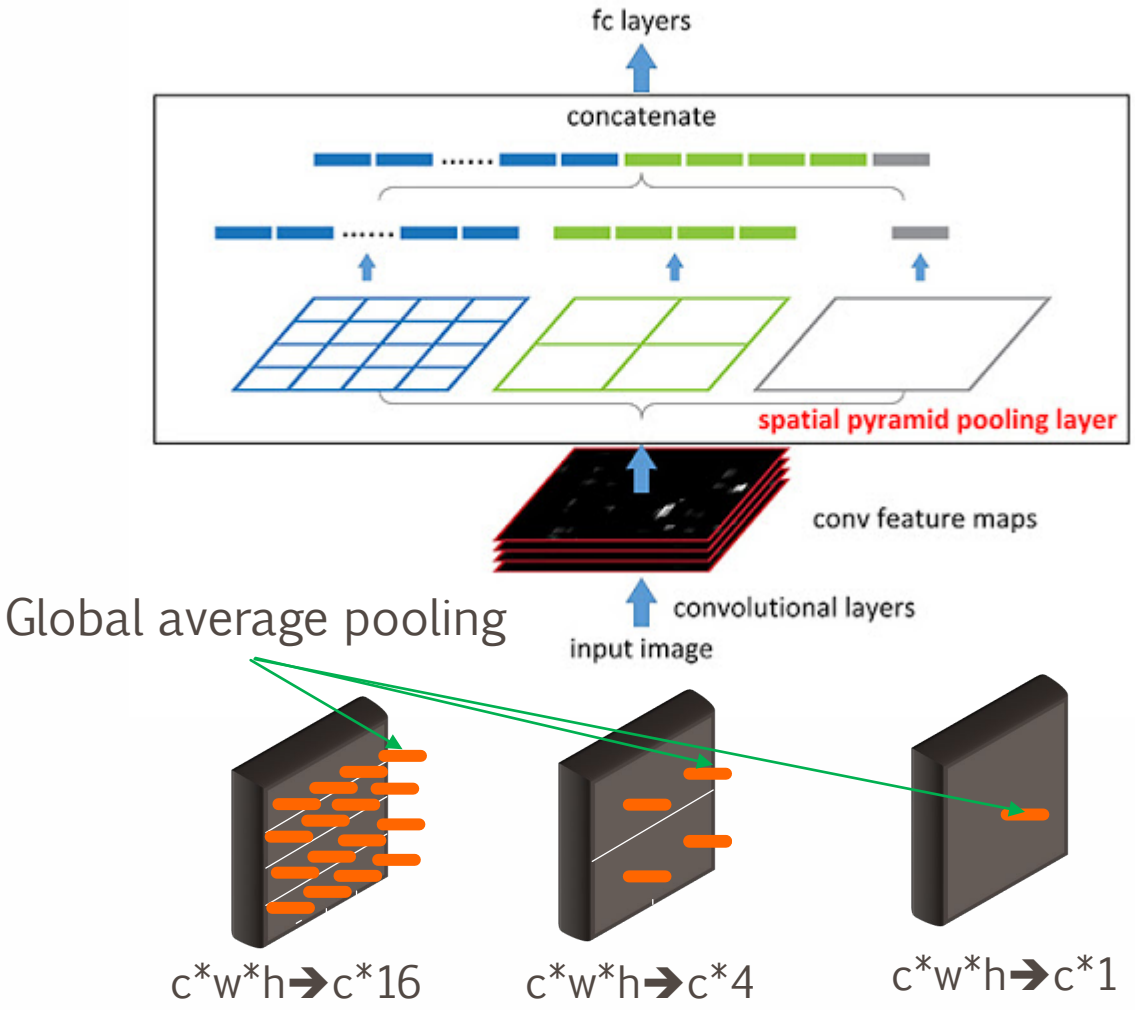


$F_{i+1} = F_i *_{2^i} k_i \quad i = 0,1,2$ $F_{i+1} = F_i *_{2^i} k_i \quad \text{for } i = 0, 1, \ldots, n-2$

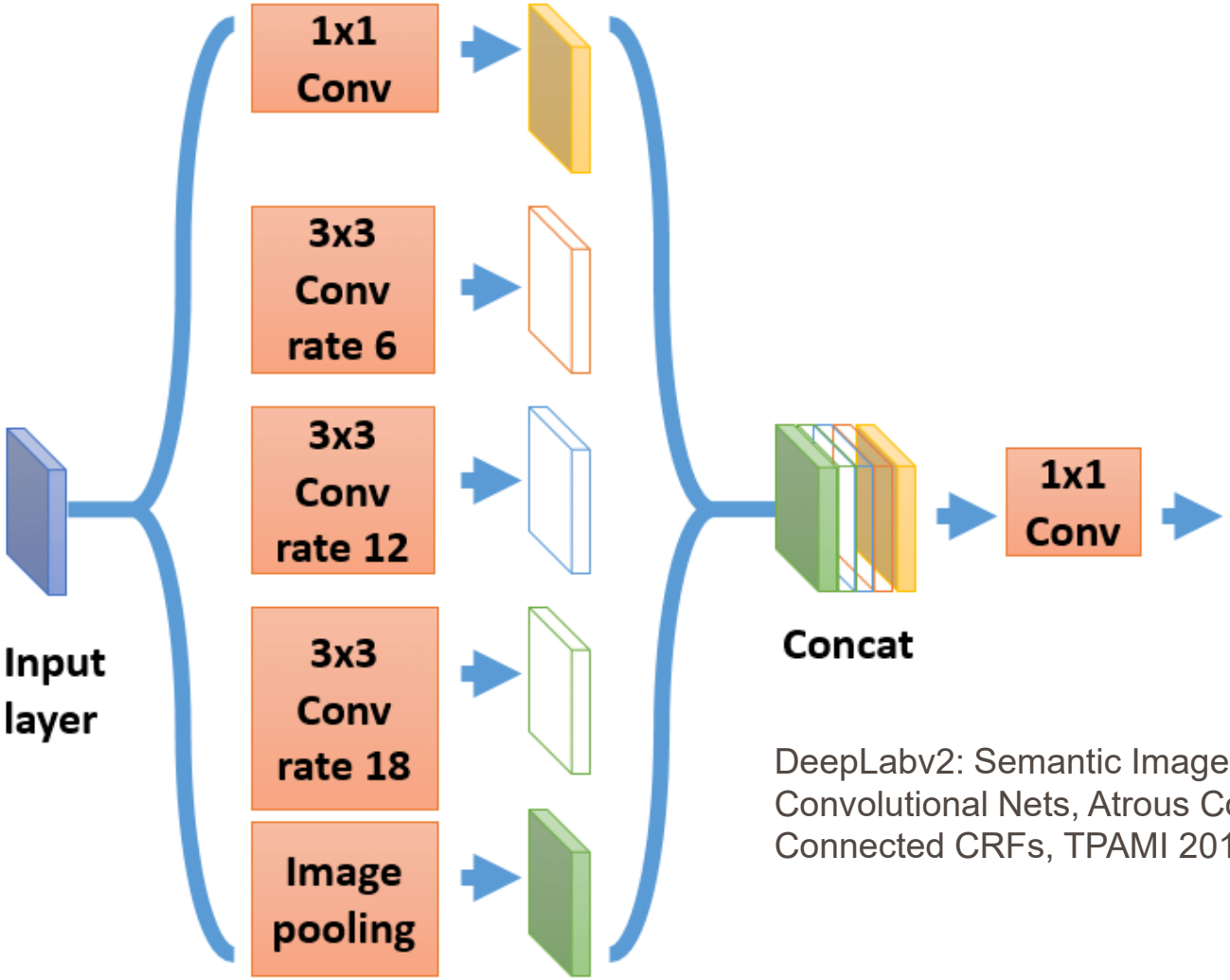
Yu, Koltun, Multi-scale context aggregation by dilated convolutions, 2016

26

SPP: Spatial Pyramid Pooling

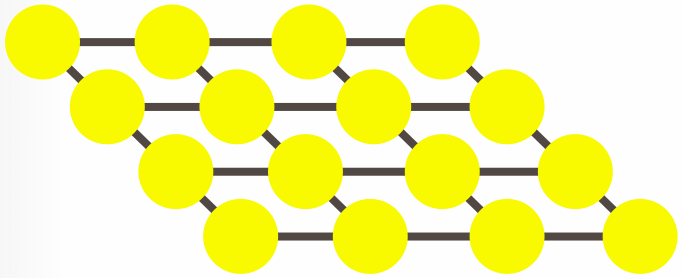


ASPP: Atrous SPP



DeepLabv2: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, TPAMI 2017

Refine: Conditional random fields



$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{y},\mathbf{x})}$$

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$$

$$= \arg\min_{\mathbf{y}} E(\mathbf{y},\mathbf{x})$$

$$E(\mathbf{y},\mathbf{x}) = \sum_i E_{data}(y_i,\mathbf{x}) + \sum_{i,j\in\mathcal{N}} E_{smooth}(y_i,y_j,\mathbf{x})$$

Conditional Random Fields

- Idea: take convolutional network prediction and sharpen using classic techniques
- *Conditional Random Field*

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} \sum_i E_{data}(y_i, \mathbf{x}) + \sum_{i,j \in \mathcal{N}} E_{smooth}(y_i, y_j, \mathbf{x})$$

$$E_{smooth}(y_i, y_j, \mathbf{x}) = \mu(y_i, y_j) w_{ij}(\mathbf{x})$$

Label compatibility

Pixel similarity

Inference in CRFs

- Problem: combinatorial optimization
- Variational methods: Approximate complex distribution p(y) with simple distribution q(y)
- Mean-field approximation: q(y) is independent distribution for each pixel:

$$q(\mathbf{y}) = \prod_i q_i(y_i)$$

- If N pixels and K classes, basically N K-dimensional vectors

Mean field inference

- If we can find best q, solution is highest probability output for each pixel
- Try to match p with q by minimizing *Kulback-Leibler Divergence*

$$KL(q||p) = \sum_{\mathbf{y}} q(\mathbf{y}) \log p(\mathbf{y}) - \sum_{\mathbf{y}} q(\mathbf{y}) \log q(\mathbf{y})$$

- Iterative process: in each iteration, do coordinate ascent on one $q(y_i)$

Mean field inference

- Coordinate descent on $q_i(y_i)$
- At each step, keep other pixels fixed and update one
- Each step (approximately):
  - Take current $q_j(y_j)$ on all $j \neq i$
  - Use this to compute $p(y_i|y_{-i})$ where $y_{-i} = \{y_j : j \neq i\}$
  - Set q_i to this

$$q_i \propto \mathbb{E}_{q_{-i}}[\log p(y_i|y_{-i})]$$

Fully Connected CRFs

- Typically, only adjacent pixels connected
  - Fewer connections => Easier to optimize
- Dense connectivity: every pixel connected to everything else
- Intractable to optimize except if pairwise potential takes specific form

$$E_{smooth}(y_i, y_j, \mathbf{x}) = \mu(y_i, y_j) w_{ij}(\mathbf{x})$$

$$w_{ij}(\mathbf{x}) = \sum_m w_m e^{-\|\mathbf{f}_m(i) - \mathbf{f}_m(j)\|^2}$$

Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. Philipp Krahenbuhl, Vladlen Koltun. In *NIPS,* 2011.

Gaussian edge potentials

- What should **f** be?
- simple answer: color, position

$$w_{ij}(\mathbf{x}) = \sum_m w_m e^{-\|\mathbf{f}_m(i) - \mathbf{f}_m(j)\|^2}$$

Mean field inference for Dense-CRF

$$q_i(y_i = l) \propto \exp[-\psi_u(y_i) - \sum_{l'} \mu(l, l') \sum_m w_m \sum_{j \neq i} e^{-\|\mathbf{f}_m(i) - \mathbf{f}_m(j)\|^2} q_j(y_j = l')]$$

Unary

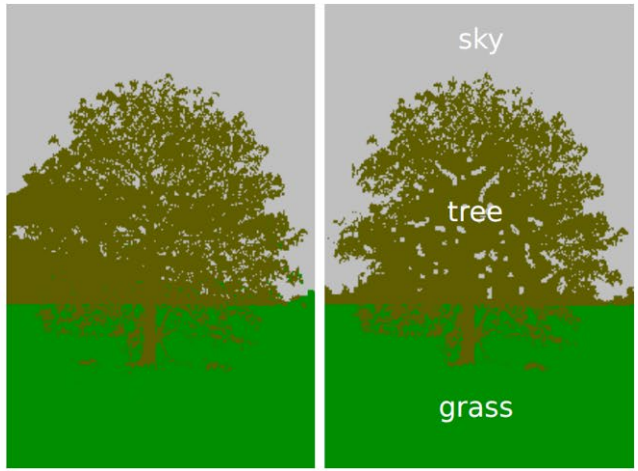
Label compatibility transform

Message passing

$$\mathbf{q}_i \propto \exp[-\boldsymbol{\psi}_u^{(i)} - \boldsymbol{\mu} \sum_j \mathbf{m}_{j \to i}]$$

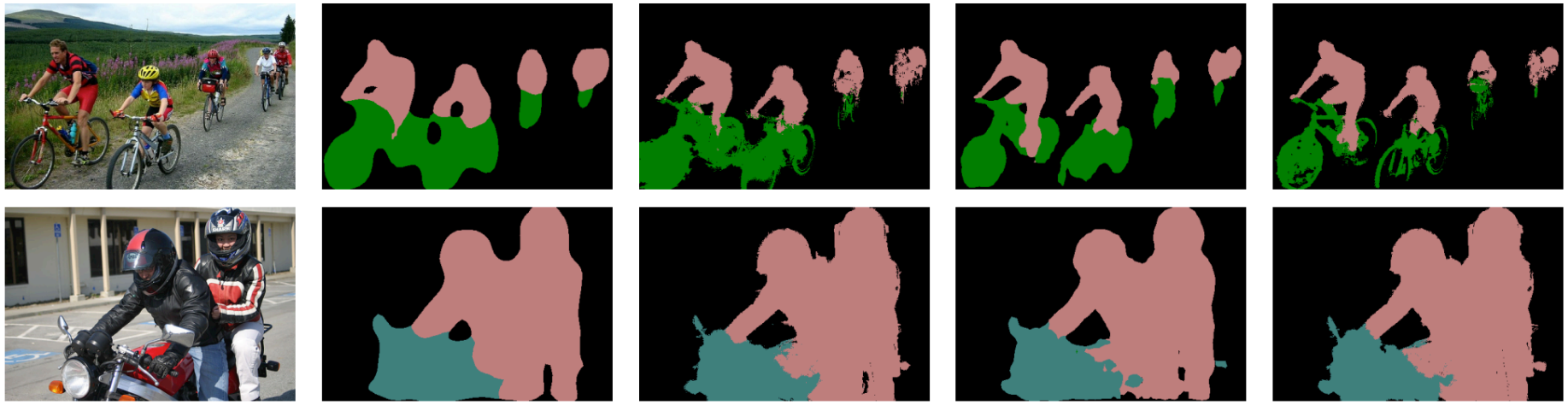
Fully Connected CRFs

Grid CRF



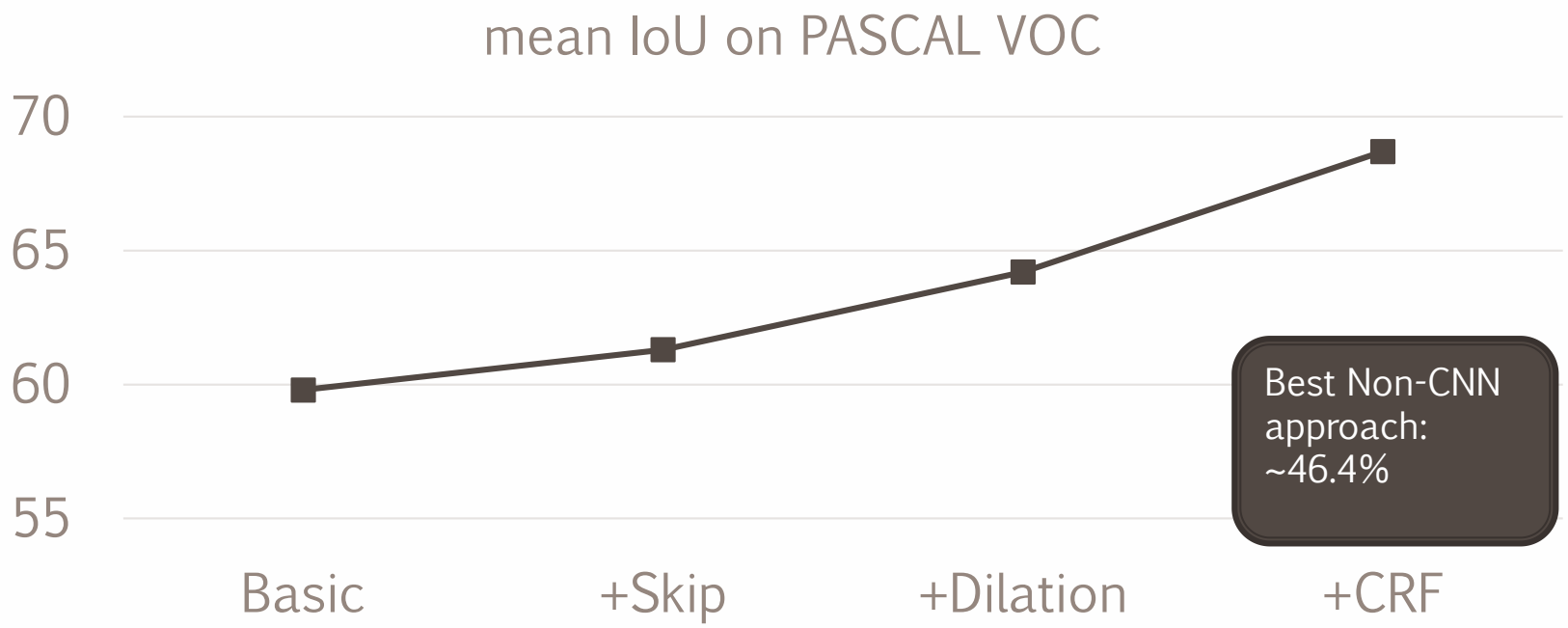
Fully
connected CRF

Fully connected CRFs



Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan Yuille. In *ICLR,* 2015.

Putting it all together



Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan Yuille. In *ICLR,* 2015.

Other additions

Method	mean IoU (%)
VGG16 + Skip + Dilation	65.8
ResNet101	68.7
ResNet101 + Pyramid	71.3
ResNet101 + Pyramid + COCO	74.9

DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan Yuille. Arxiv 2016.

Mean field inference as a recurrent network

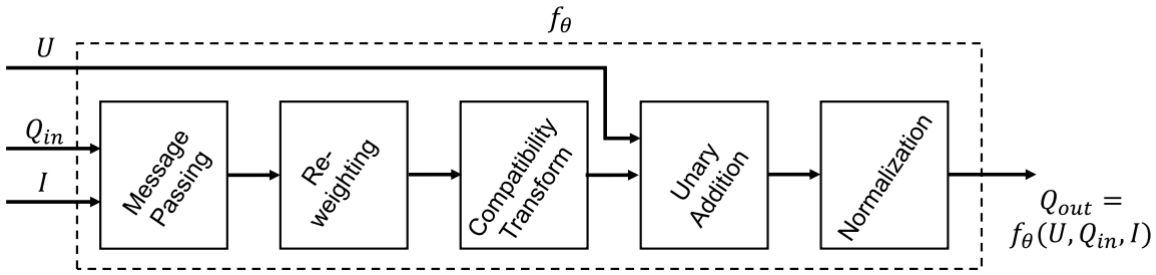
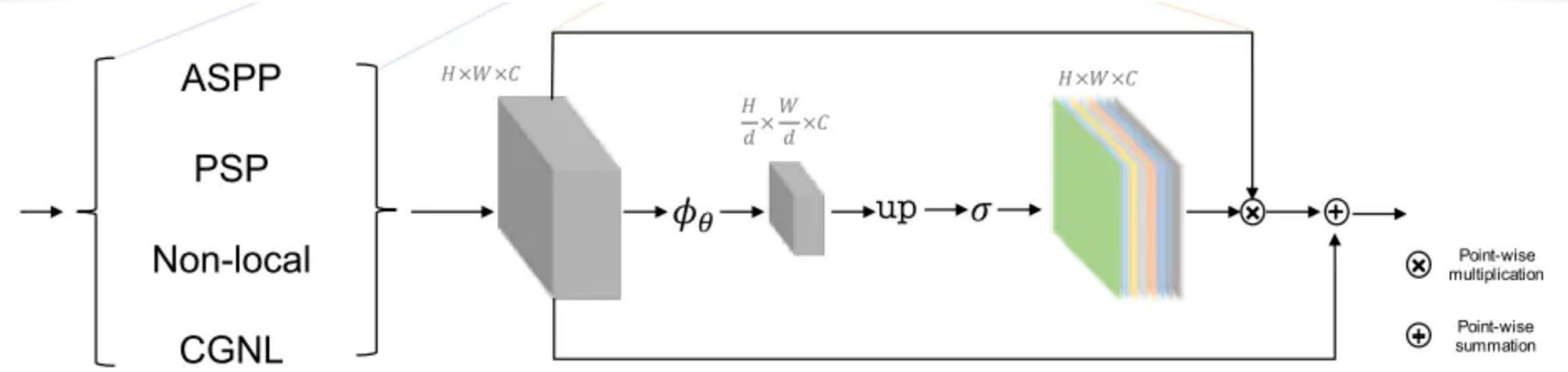


Figure 1. **A mean-field iteration as a CNN.** A single iteration of the mean-field algorithm can be modelled as a stack of common CNN layers.

Zheng, Shuai, et al. "Conditional random fields as recurrent neural networks." *Proceedings of the IEEE international conference on computer vision*. 2015.

Semantic Segmentation: A summary

- The common structure
- Contextual information mining + segmentation head



SEGMENTATION TRANSFORMER: OBJECT-CONTEXTUAL REPRESENTATIONS FOR SEMANTIC SEGMENTATION

ECCV2020

OCR

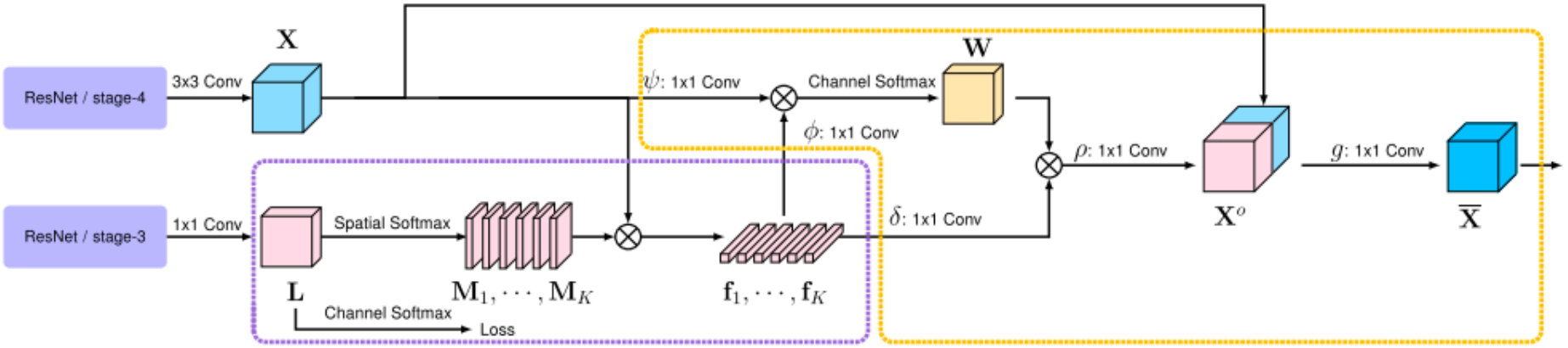
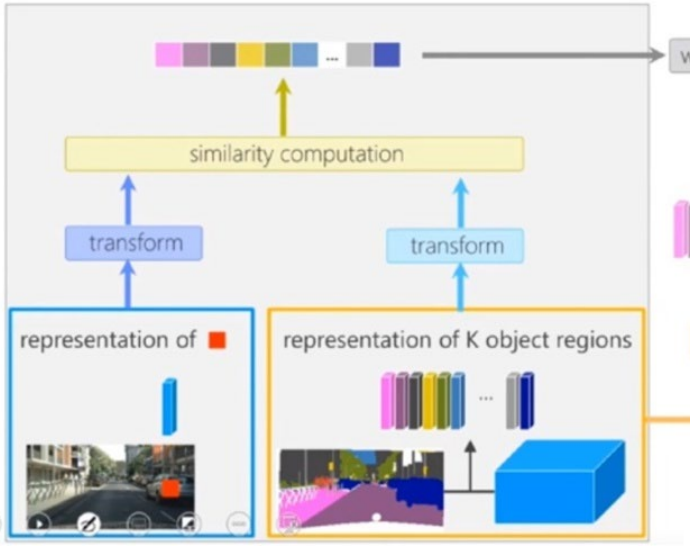


Figure 2: **The pipeline of ou** ...entation in the *purple dashed box* and compute the object co... element of **X**) and **L** into the *purple dashed box* to compute ...ations and **X** into the *orange dashed box* to compute the obje... utput from stage-4 of ResNet and stage-3 of ResNet separate... ns of all the representations/ feature maps.



Good Backbone makes good results

- Precise information required
  - High-resolution network !

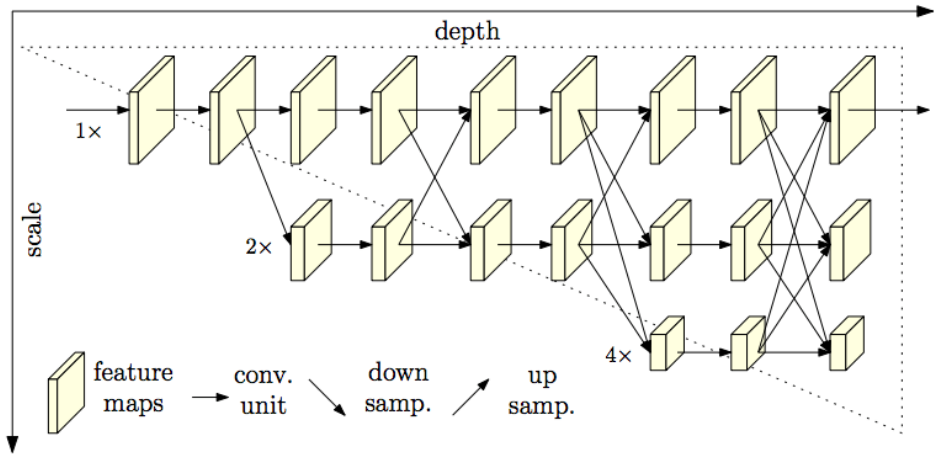


Figure 1. Illustrating the architecture of the proposed HRNet. It consists of parallel high-to-low resolution subnetworks with repeated information exchange across multi-resolution subnetworks (multi-scale fusion). The horizontal and vertical directions correspond to the depth of the network and the scale of the feature maps, respectively.

Ke Sun, Bin Xiao, Dong Liu, Jingdong Wang. "Deep High-Resolution Representation Learning for Human Pose Estimation". In CVPR'19.

HRNET vs. Traditional Backbone

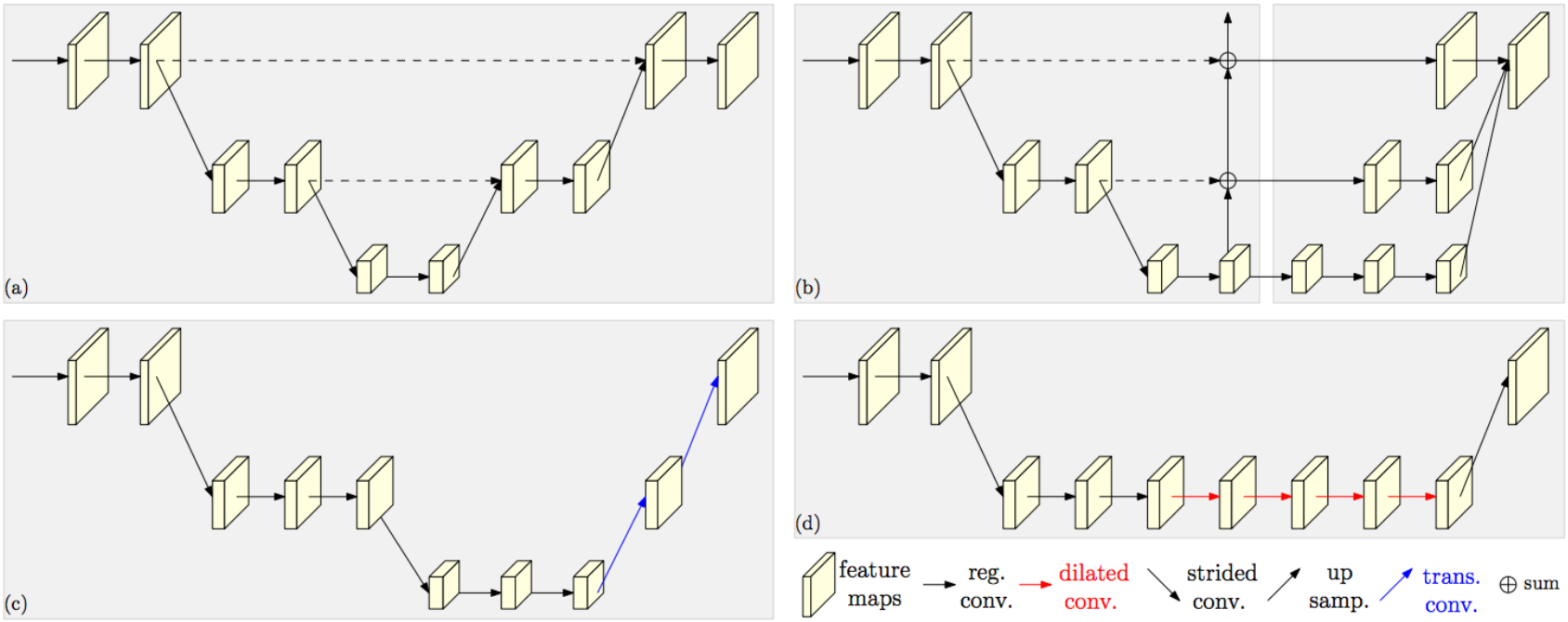
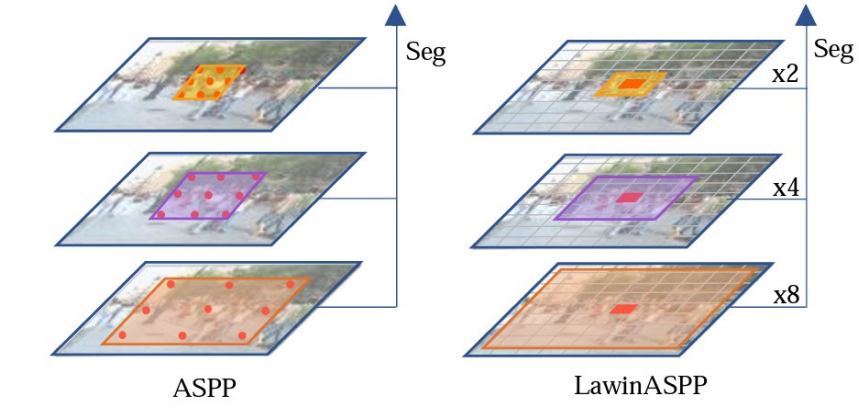
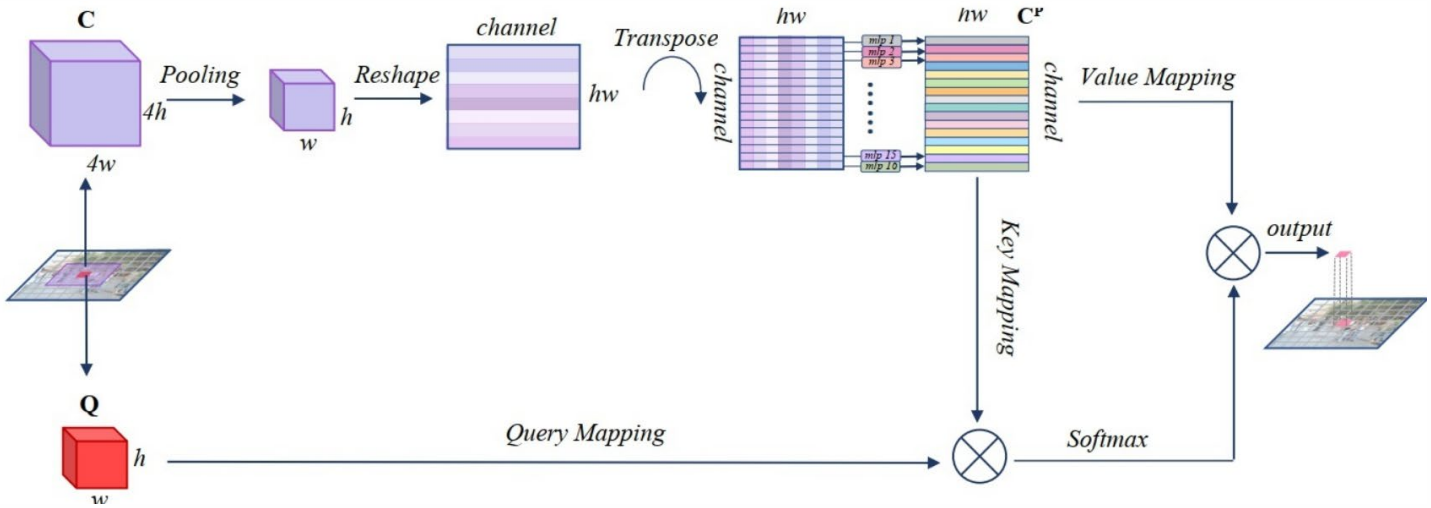


Figure 2. Illustration of representative pose estimation networks that rely on the high-to-low and low-to-high framework. (a) Hourglass [40]. (b) Cascaded pyramid networks [11]. (c) SimpleBaseline [72]: transposed convolutions for low-to-high processing. (d) Combination with dilated convolutions [27]. Bottom-right legend: reg. = regular convolution, dilated = dilated convolution, trans. = transposed convolution, strided = strided convolution, concat. = concatenation. In (a), the high-to-low and low-to-high processes are symmetric. In (b), (c) and (d), the high-to-low process, a part of a classification network (ResNet or VGGNet), is *heavy*, and the low-to-high process is *light*. In (a) and (b), the skip-connections (dashed lines) between the same-resolution layers of the high-to-low and low-to-high processes mainly aim to fuse low-level and high-level features. In (b), the right part, refinenet, combines the low-level and high-level features that are processed through convolutions.

SOTA (2022): Lawin Transformer



- Larger kernel improves the performance



Lawin Transformer: Improving Semantic Segmentation Transformer with Multi-Scale Representations via Large Window Attention, 2022

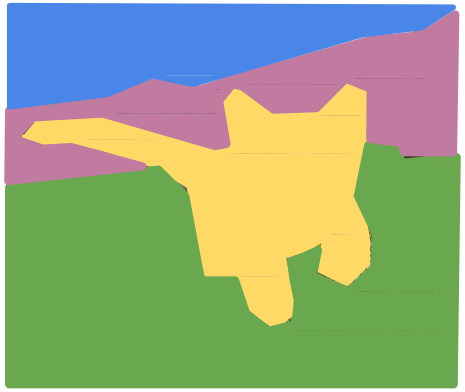
Recap: Lots of computer vision tasks!

Classification

CAT

No spatial extent

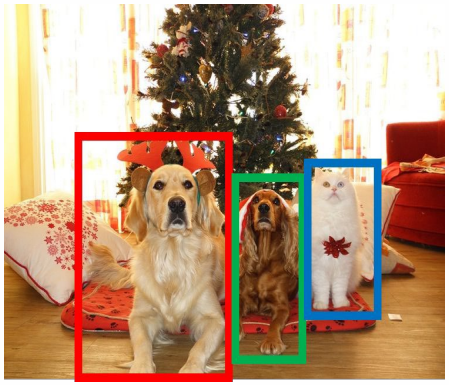
Semantic Segmentation



GRASS, CAT, TREE, SKY

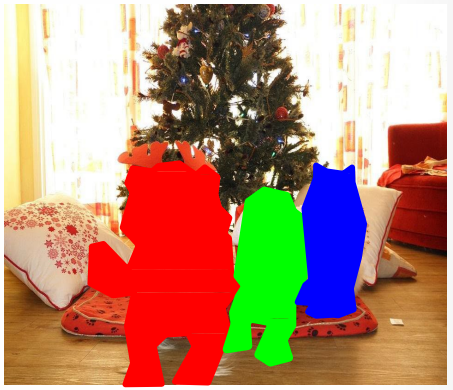
No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

Instance Segmentation

Classification

CAT

No spatial extent

Semantic Segmentation

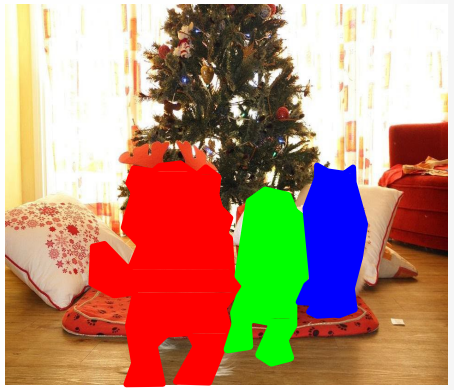
GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection

DOG, DOG, CAT

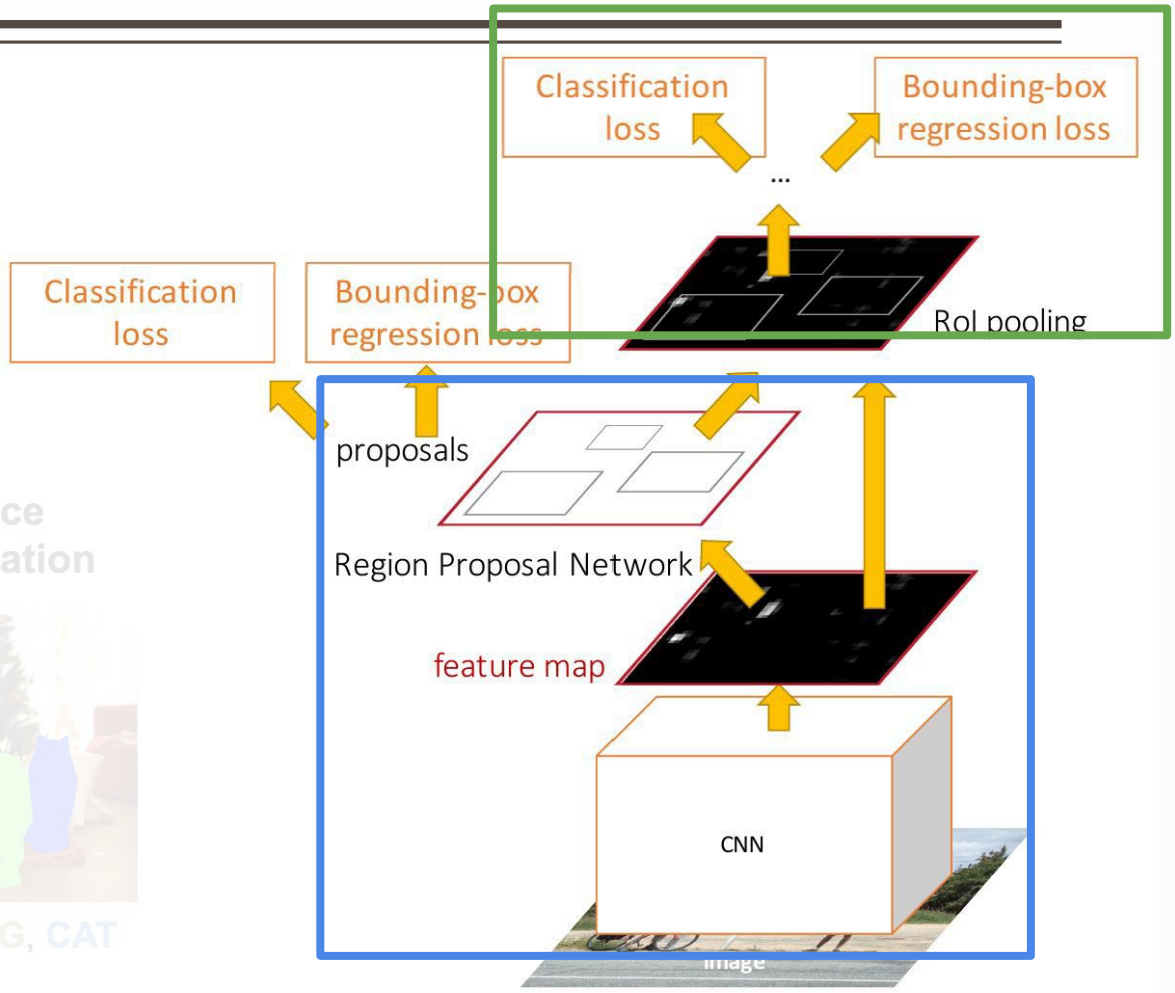
Instance Segmentation



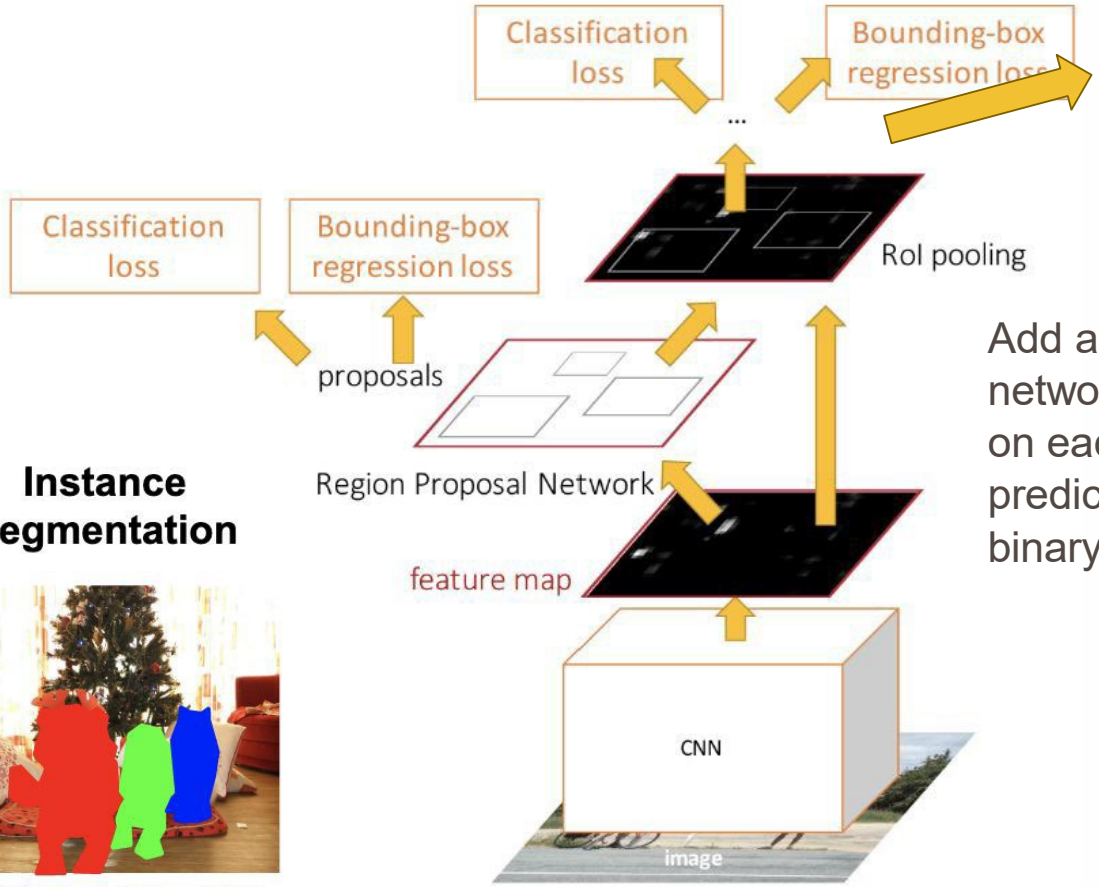
DOG, DOG, CAT

Multiple Object

Object Detection: Faster R-CNN



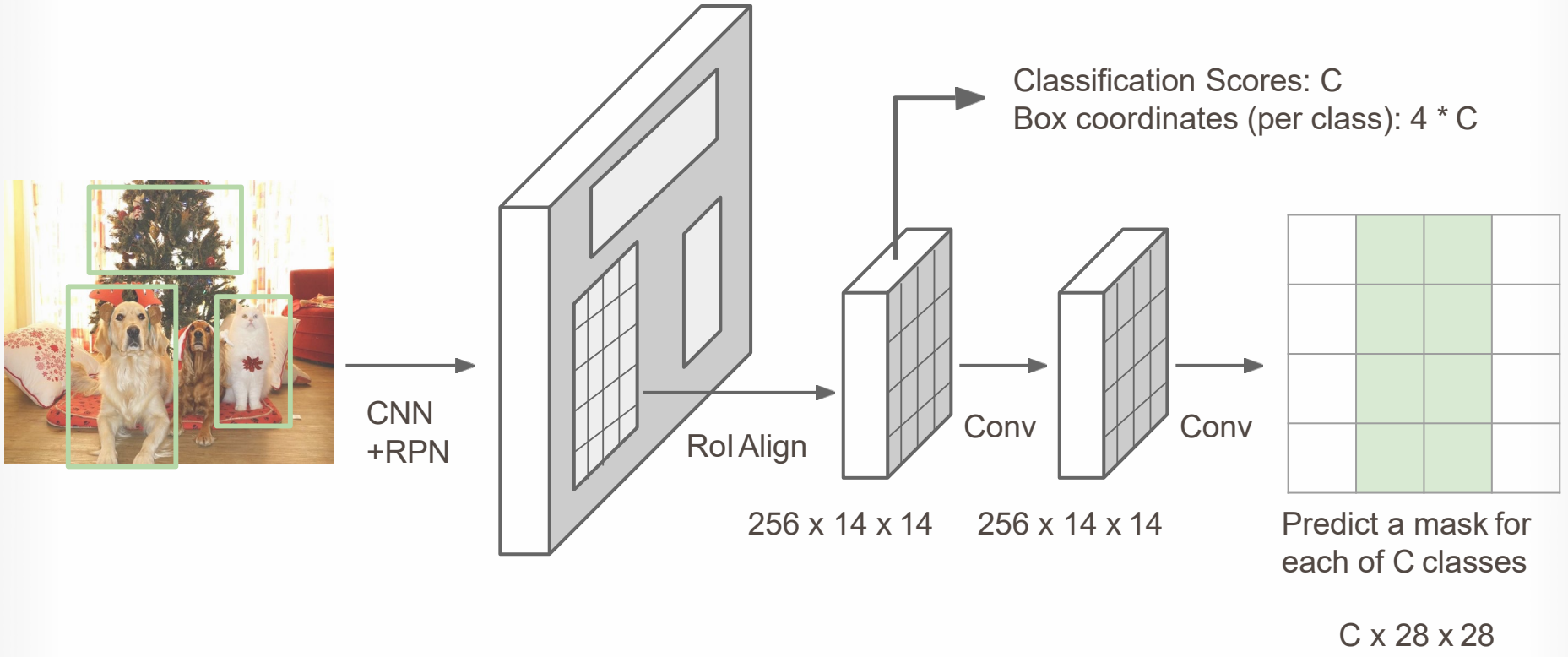
Instance Segmentation: Mask R-CNN



Add a small mask network that operates on each RoI and predicts a 28x28 binary mask

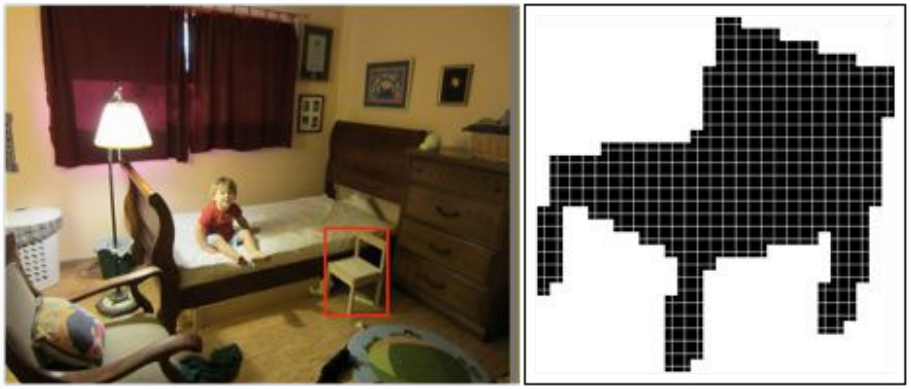
He et al, "Mask R-CNN", ICCV 2017

Mask R-CNN

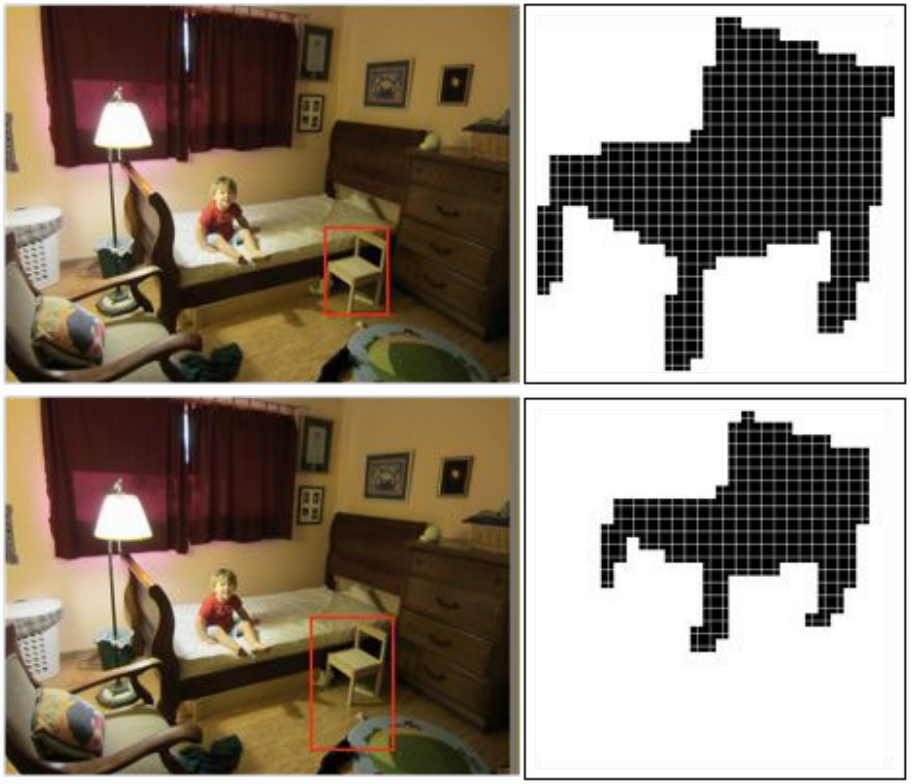


He et al, “Mask R-CNN”, arXiv 2017

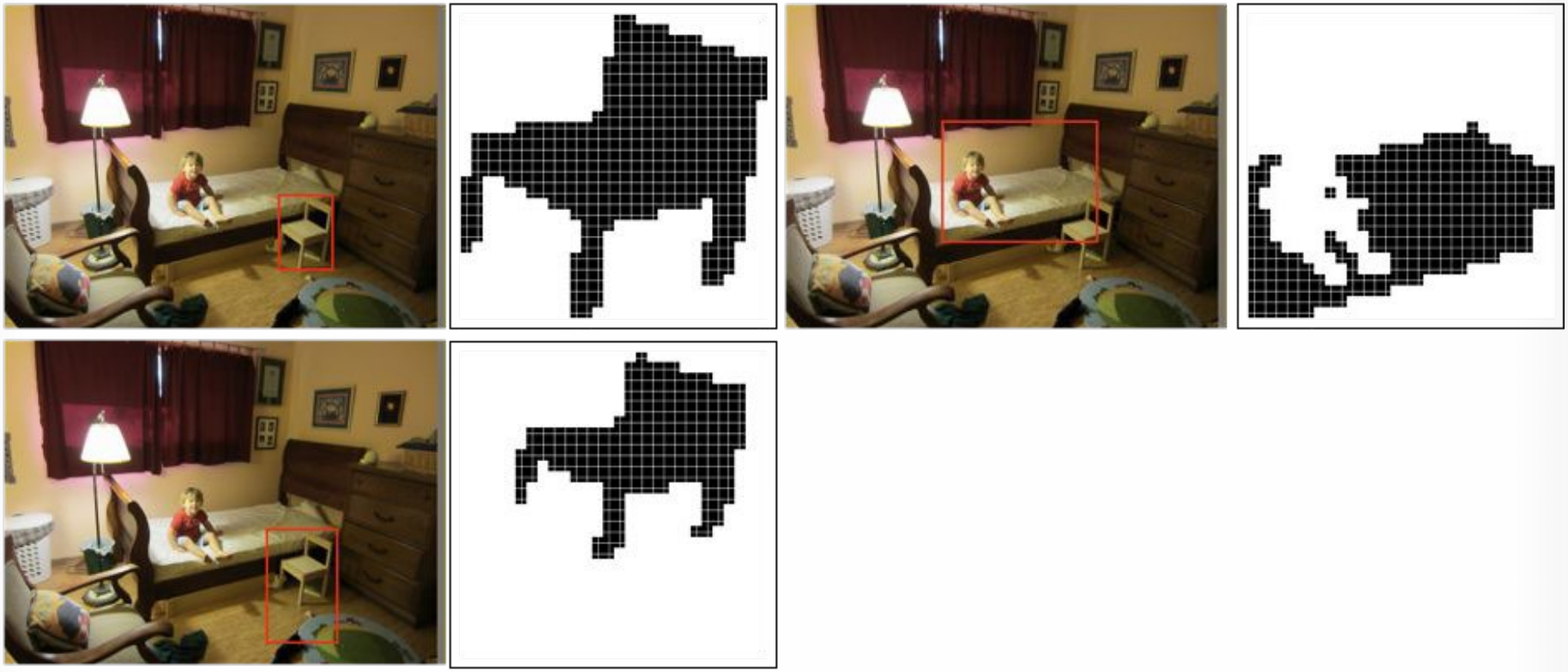
Mask R-CNN: Example Mask Training Targets



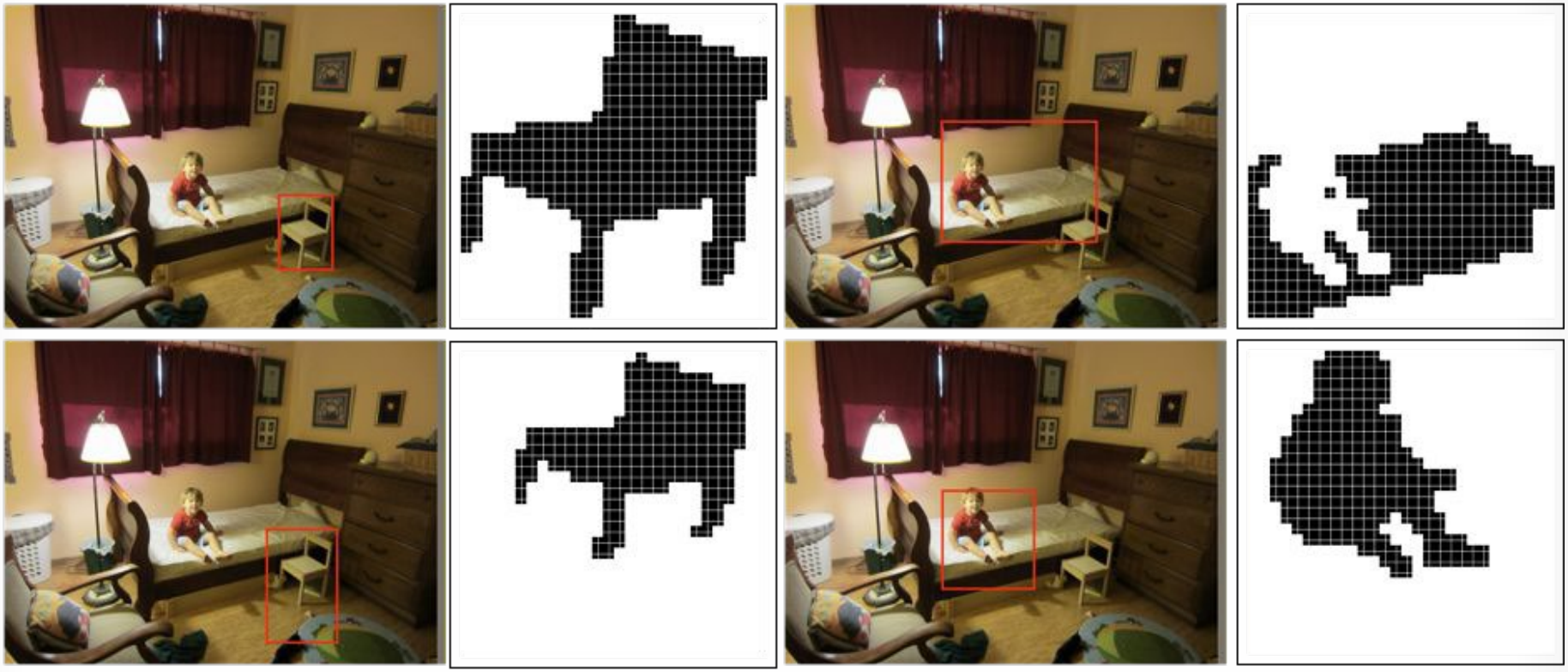
Mask R-CNN: Example Mask Training Targets



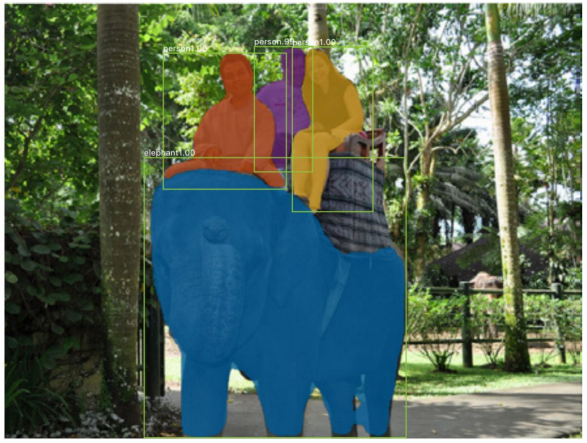
Mask R-CNN: Example Mask Training Targets

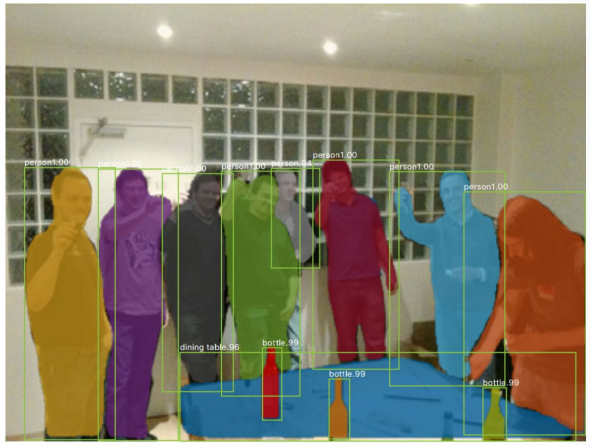


Mask R-CNN: Example Mask Training Targets



Mask R-CNN: Very Good Results!





He et al, “Mask R-CNN”, ICCV 2017

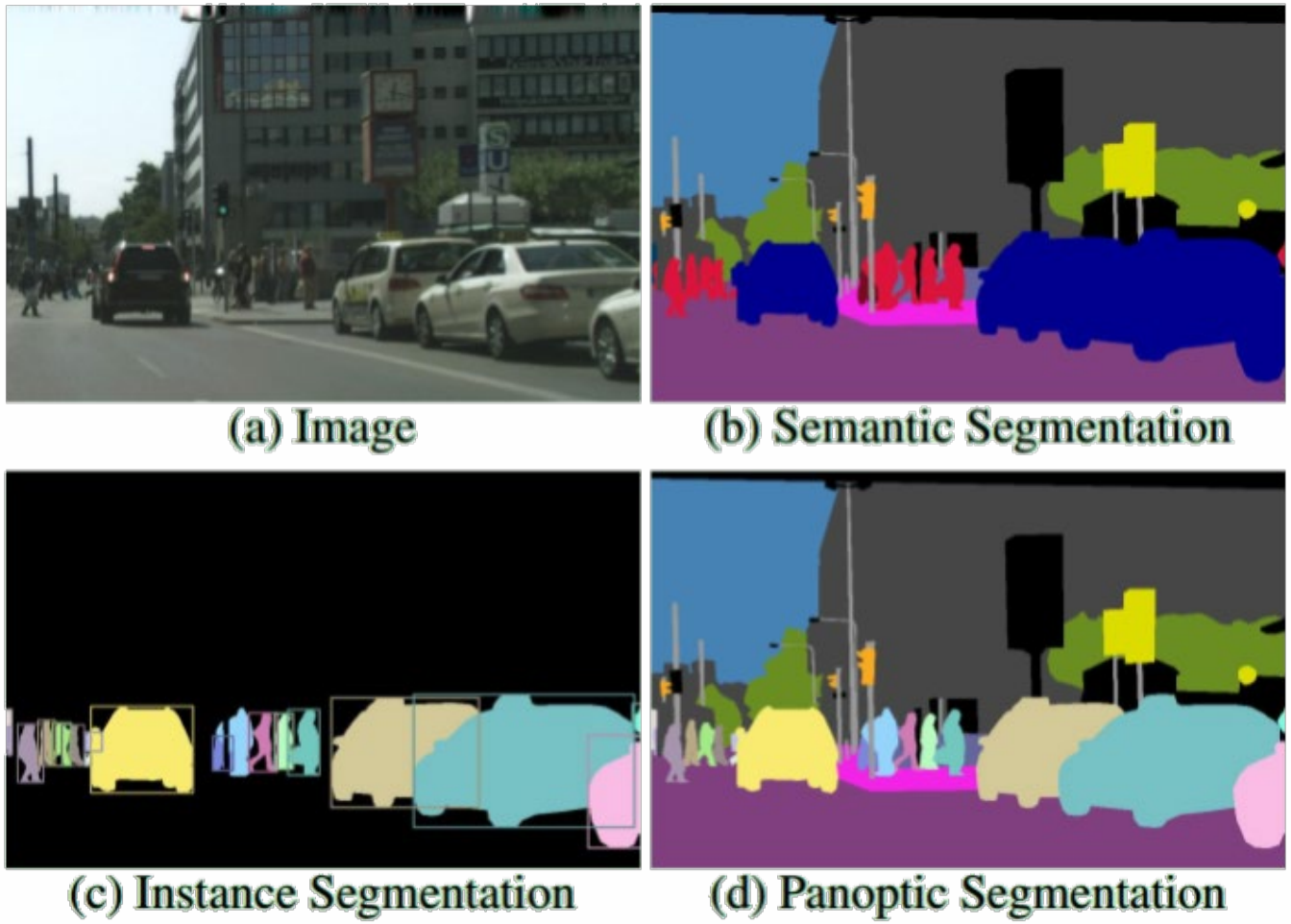
Mask R-CNN Also does pose

He et al, “Mask R-CNN”, ICCV 2017

LPSNET: A LIGHTWEIGHT SOLUTION FOR FAST PANOPTIC SEGMENTATION (CVPR21)

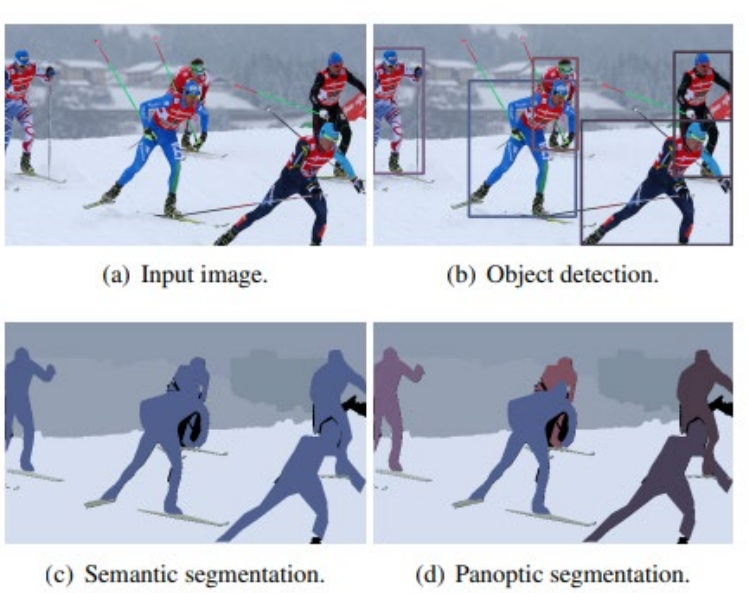
A fast approach

What is “Panoptic segmentation”?



Introduction - Panoptic segmentation

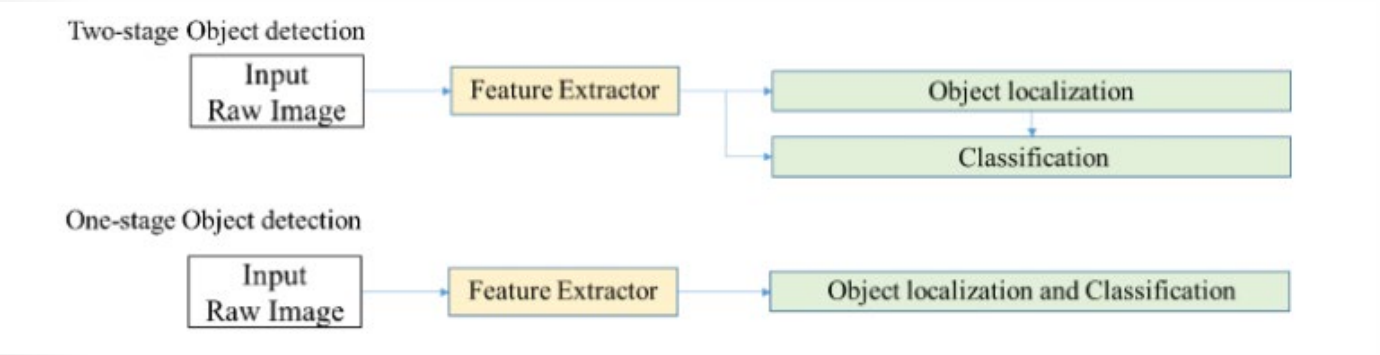
- Panoptic segmentation
  - Object detection
  - Semantic segmentation - Stuff
  - Instance segmentation - Thing
- Multi-Task Learning(MTL)



(a) Input image. (b) Object detection.

(c) Semantic segmentation. (d) Panoptic segmentation.

Introduction – Object detection



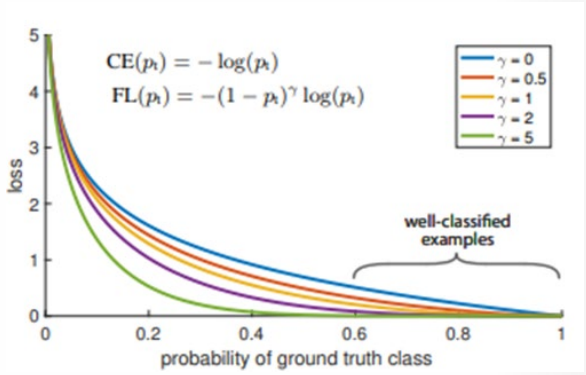
Region Proposal Network

	Benefits	Drawbacks
Two-stage	Higher accuracy	1. Heavy in resources consumption and slow in inference 2. Conflicts with the output from semantic segmentation branch, hence heuristic or complex post-process
One-stage	Faster	Lower accuracy

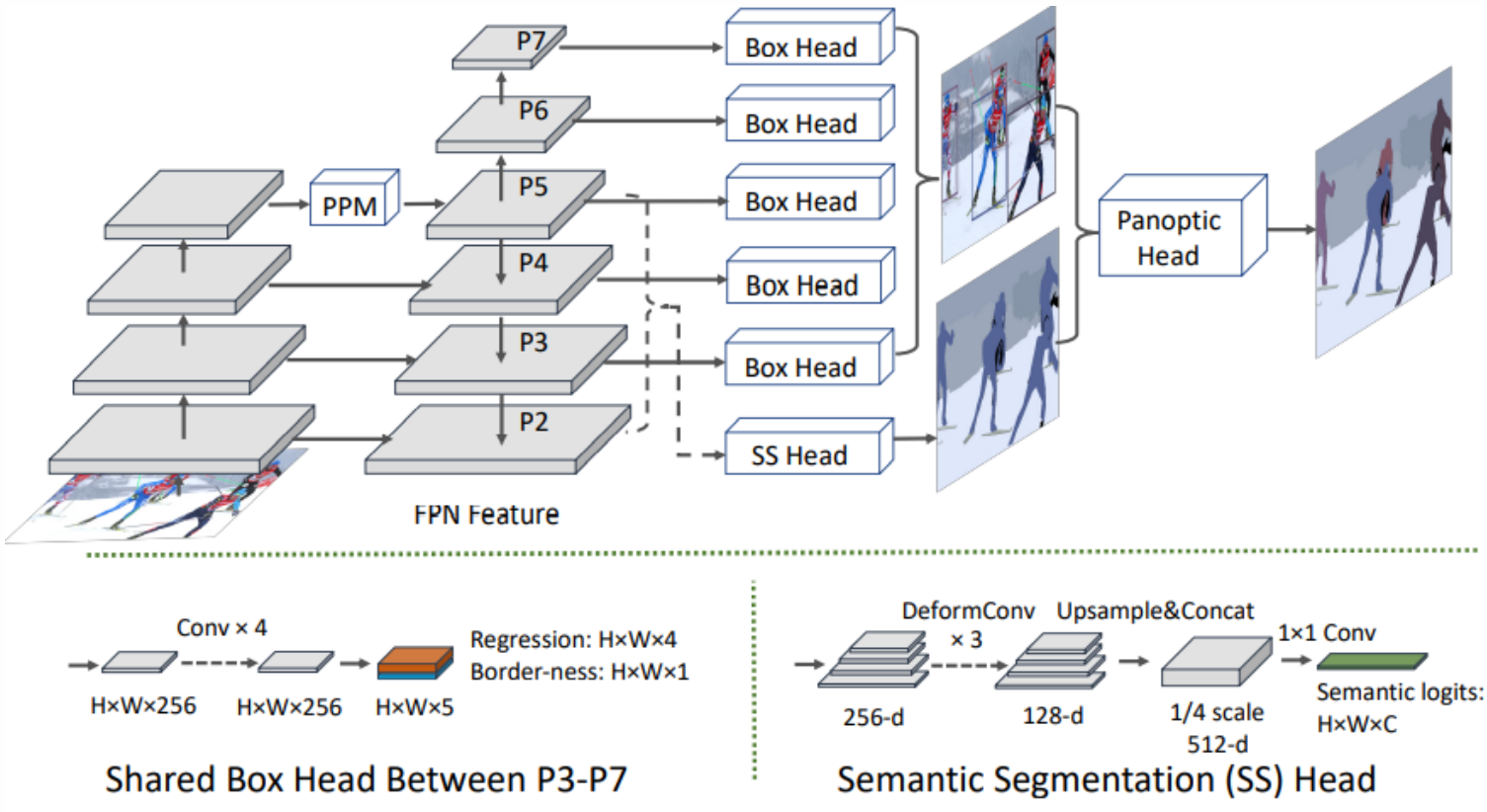
Introduction - Anchor

- Anchor-based
  - Drawbacks:
    - 1. Hyper-parameters of anchor settings need carefully tuned.
    - 2. The computation and storage costs related to anchor boxes are also heavy.

- Anchor-free
  - Drawback:
    - 1. Imbalance between positive and negative examples
    - and also between hard and easy examples -> Focal Loss
    - 2. Difficulties on different scale -> Feature Pyramid Network

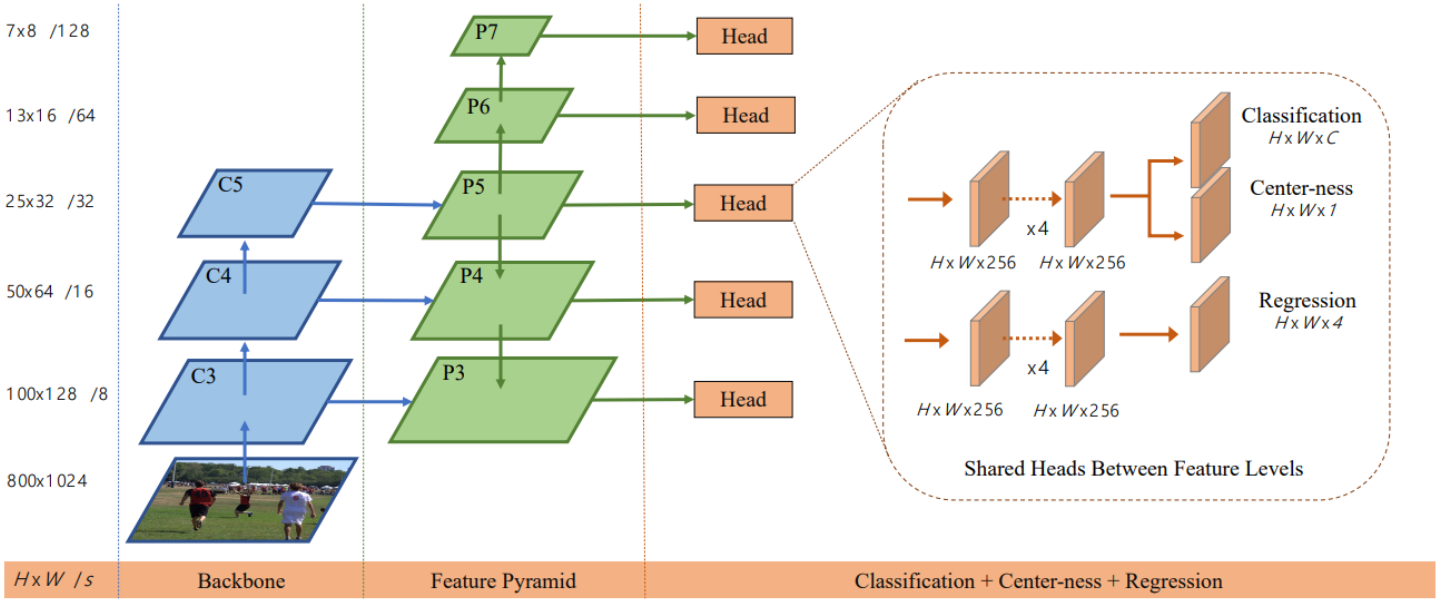


LPSnet - Model



Backbone - FCOS

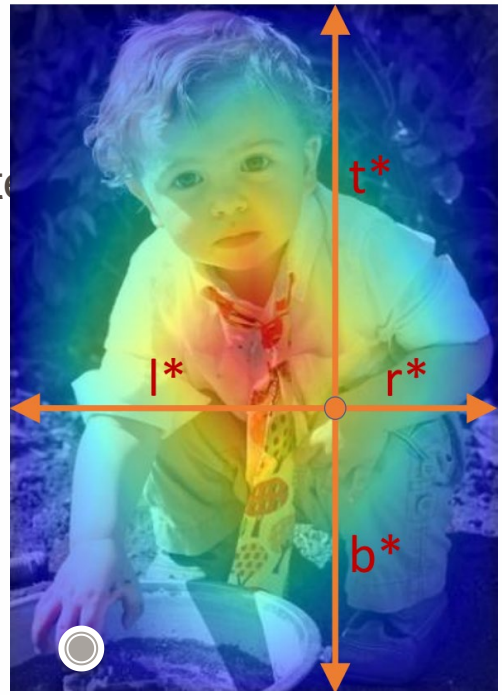
- Fully Convolutional One-Stage Object Detection



Backbone - Head box of FCOS

- Works
  - Classification - One hot vector output
  - Regression - [l*, t*, r*, b*]
  - Center-ness - Normalized distance from cente
- Center - ness
  - Non-maximum suppression
  - Filter out low-quality predicted bounding

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}.$$



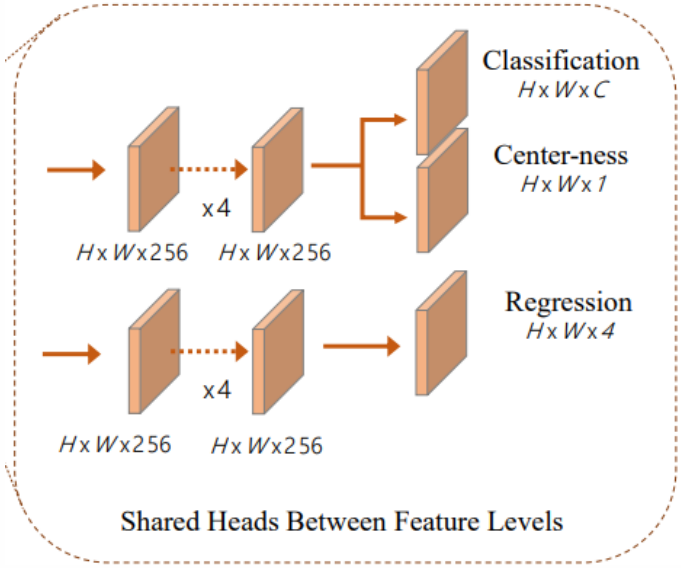
Backbone - Head box of FCOS

- Loss Function (Focal loss & IOU loss & BCE)

$$L(\{\mathbf{p}_{x,y}\},\{\mathbf{t}_{x,y}\}) = \frac{1}{N_{\text{pos}}}\sum_{x,y} L_{\text{cls}}(\mathbf{p}_{x,y}, c^*_{x,y}) + \frac{1}{N_{\text{pos}}}\sum_{x,y}[c^*_{x,y} > 0]\cdot L_{\text{reg}}(\mathbf{t}_{x,y}, \mathbf{t}^*_{x,y}) + \frac{1}{N_{\text{pos}}}\sum_{x,y}\text{BCELoss}(\text{centerness})$$

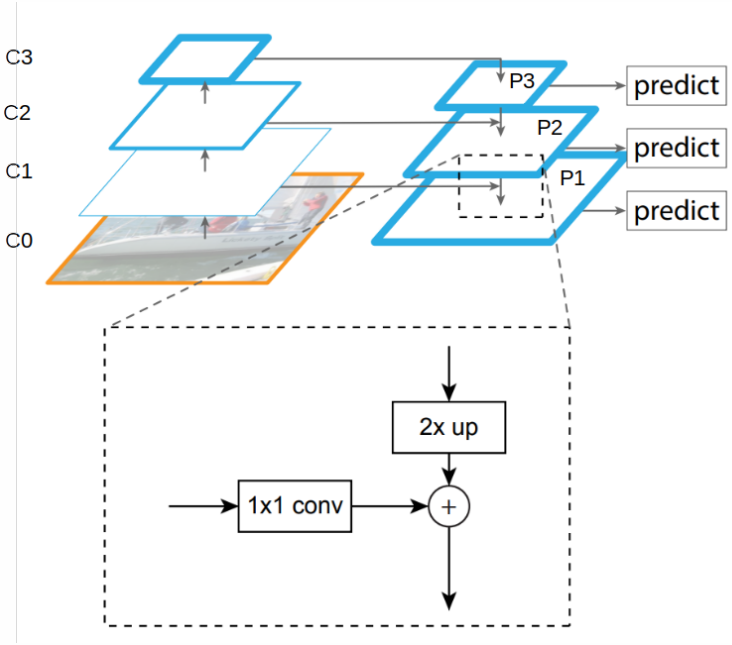
$$L_{cls}(p_t) = -\alpha(1-p_t)^r \log(p_t)$$

$$L_{reg}(t,t^*) = \frac{area\ of\ overlap(t,t^*)}{area\ of\ union(t,t^*)}$$



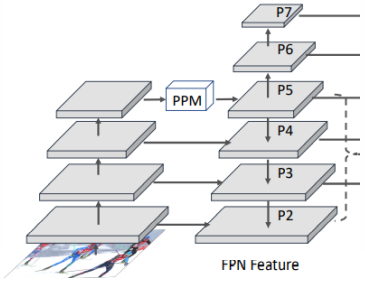
Backbone - FPN

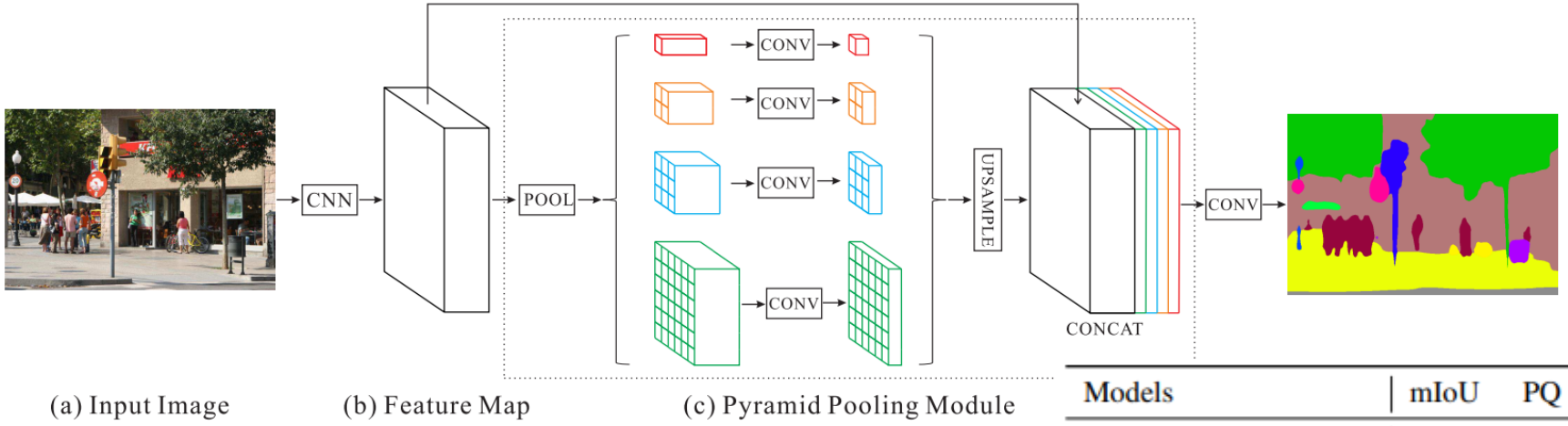
- Feature Pyramid Networks
  - Architecture
    - Down Sampling & Up Sampling
    - skip-connection (lateral connection)
    - top-down
  - Advantage
    - Usage of multi-scale feature map
    - Deal with overlapping object



Backbone - PPM

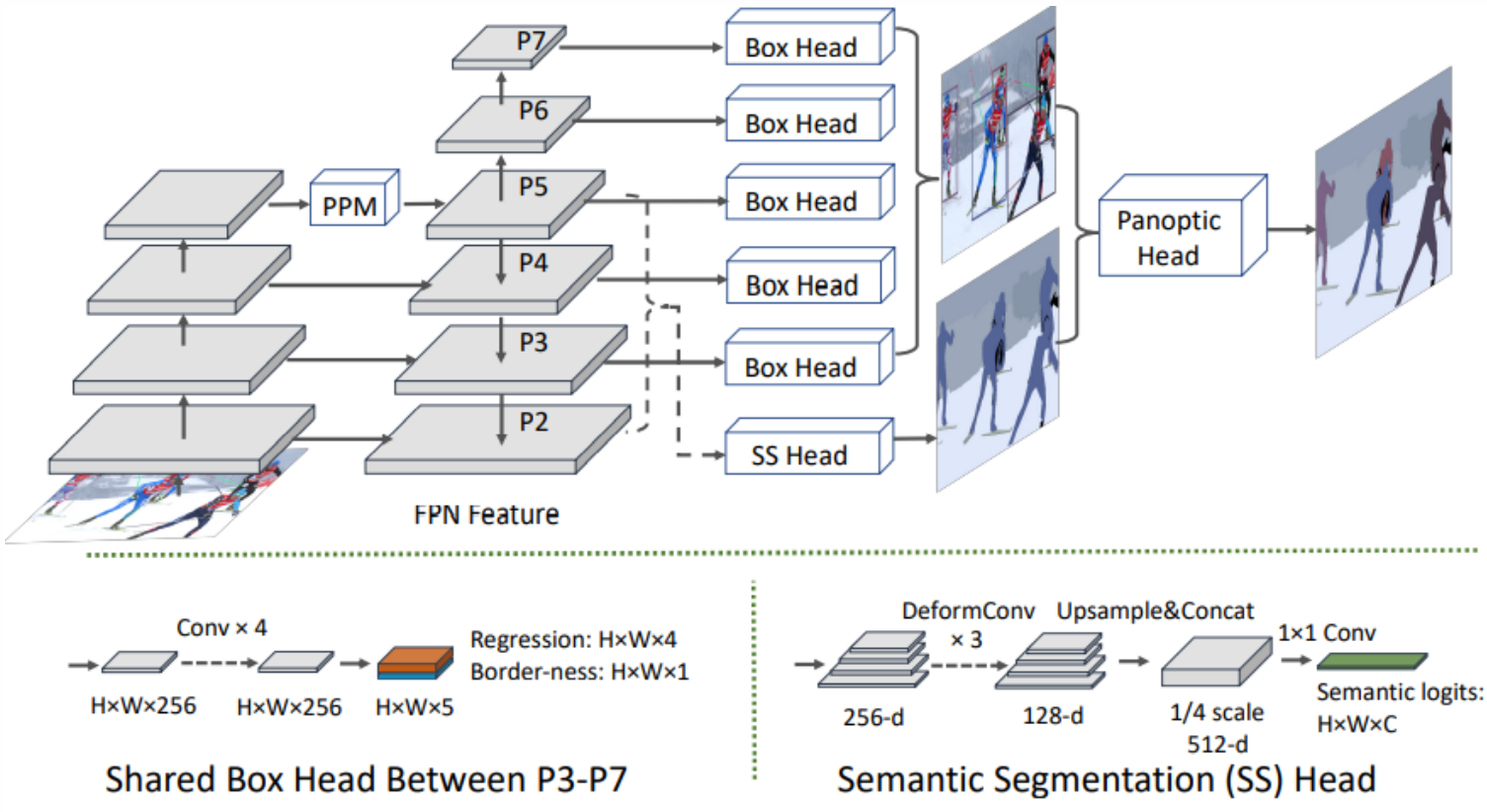
- Pyramid Scene Parsing Network
  - Usage of multi-scale feature map





Models	mIoU	PQ
Global Average Pooling	51.2	36.5
Without PPM	50.9	35.7
With PPM	54.5	39.1

LPSnet - Model



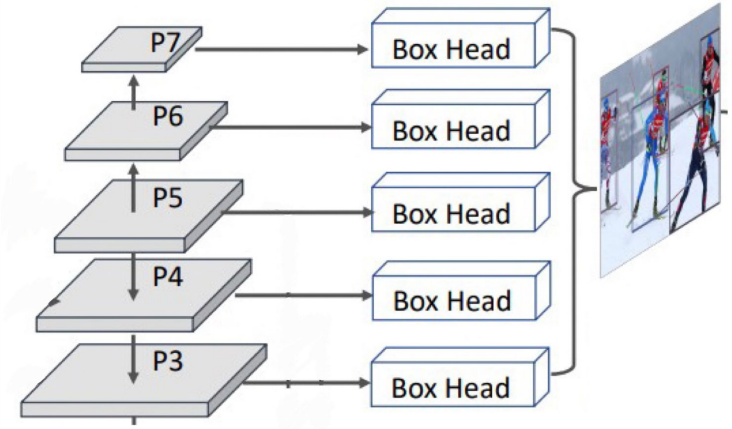
Box Head

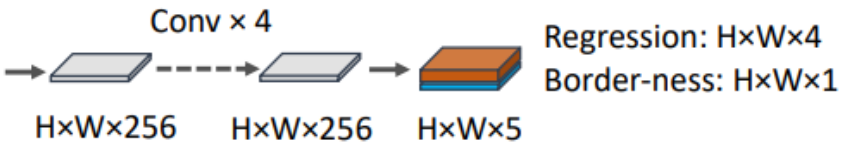
- 功能: objecct detection
- input: P3-P7 H×W×256
- output: H × W × 5
- [l, t, r, b, p]
- p: Border-ness

(b) Object detection.

Box Head

- takes P3-P7 feature as input
- A point is considered as a positive sample if it is part of ground-truth mask of things



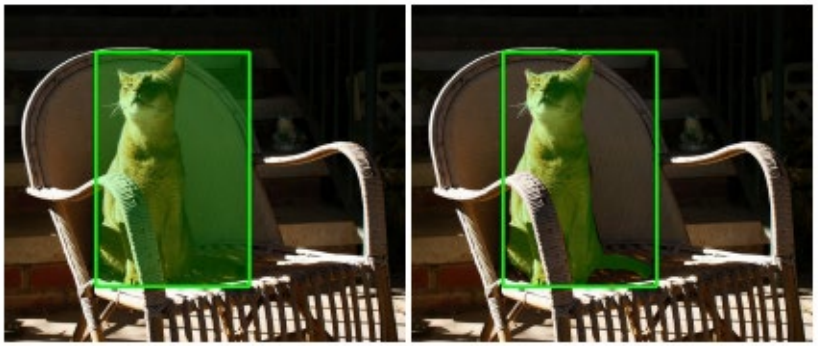


Shared Box Head Between P3-P7

Box Head

- mask-based positive sample
- border-ness
  - trained with binary cross entropy (BCE) los
- non-maximum suppression (NMS)

$$\text{border-ness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}$$

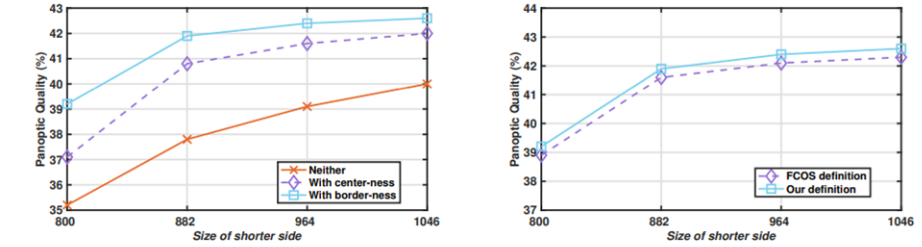


(a) FCOS [31]. (b) Ours.

(a) Center-ness [31]. (b) Border-ness.

Border-ness & mask-based positive sample

Strategies	PQ	PQ^{Th}	PQ^{St}
Smallest Area	38.9	43.5	30.0
Highest Confidence	39.0	43.6	30.0
Closest Border-ness	39.1	43.9	30.1

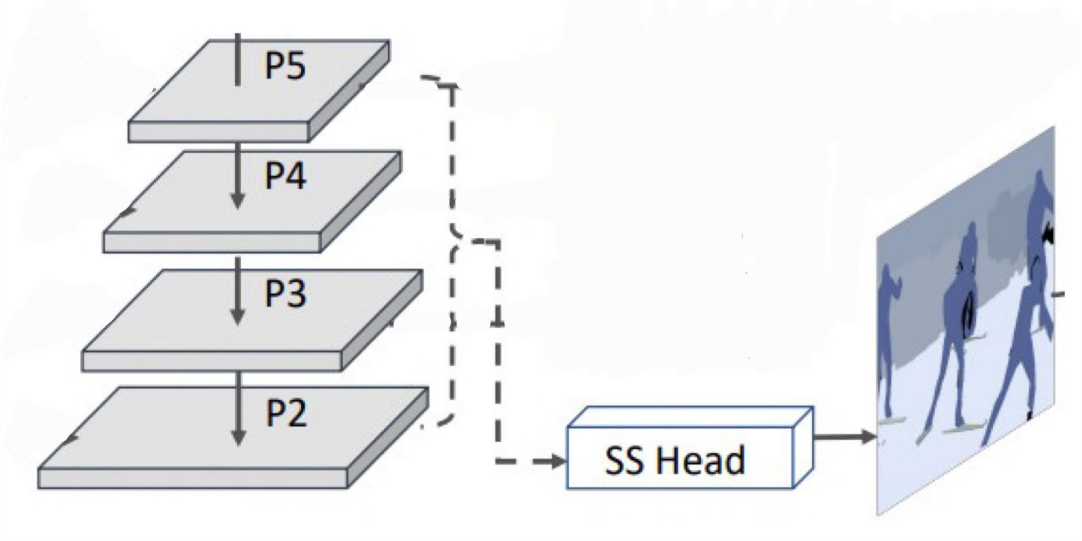


(a) Effect of border-ness. (b) Different positive sample def.

Figure 7. The experiments are conducted on COCO datset with ResNet-50 backbone. (a). Our border-ness demonstrates advantages to center-ness on various train/test image size. (b). Our definition of positive samples leads to improved performances than that of FCOS [31].

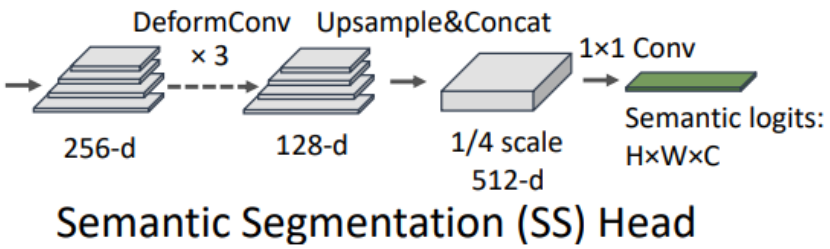
Semantic Segmentation Head

- Purpose: parse the semantic classes
- input: 256-channel P2, P3, P4 and P5 feature maps



Semantic Segmentation Head

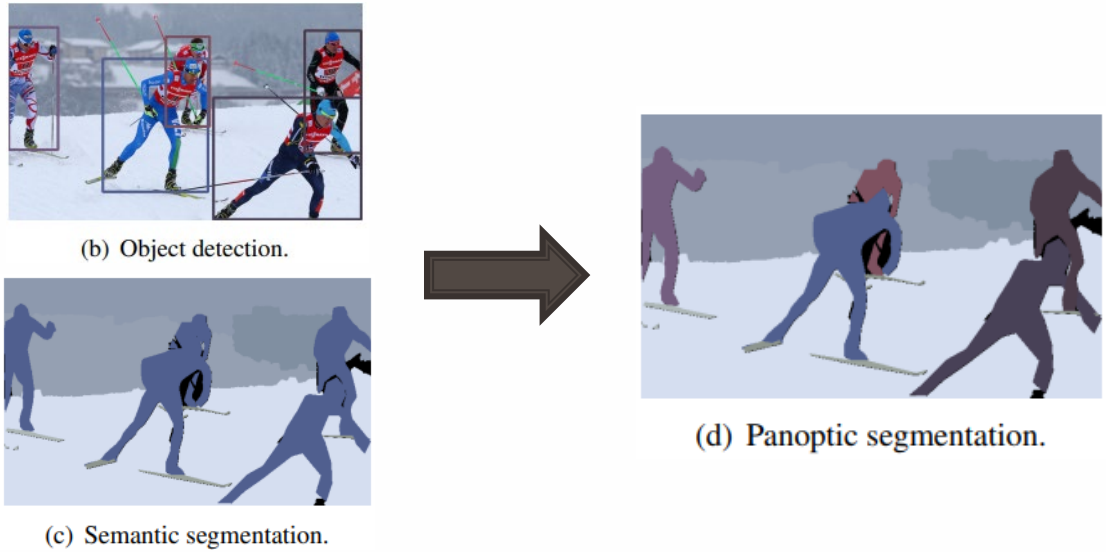
- Overall process:
  - Feature maps are processed by our deformable convolution network independently
  - upsampled to 1/4 scale of the input image by bilinear interpolation
  - concatenate them and apply 1×1 convolutions with softmax to predict the semantic class.
- pixel-wise cross entropy loss



Semantic Segmentation (SS) Head

Panoptic Head

- Purpose: produce panoptic segmentation based on bounding boxes and per-pixel semantic segmentation
- Designed as a non-parametric module



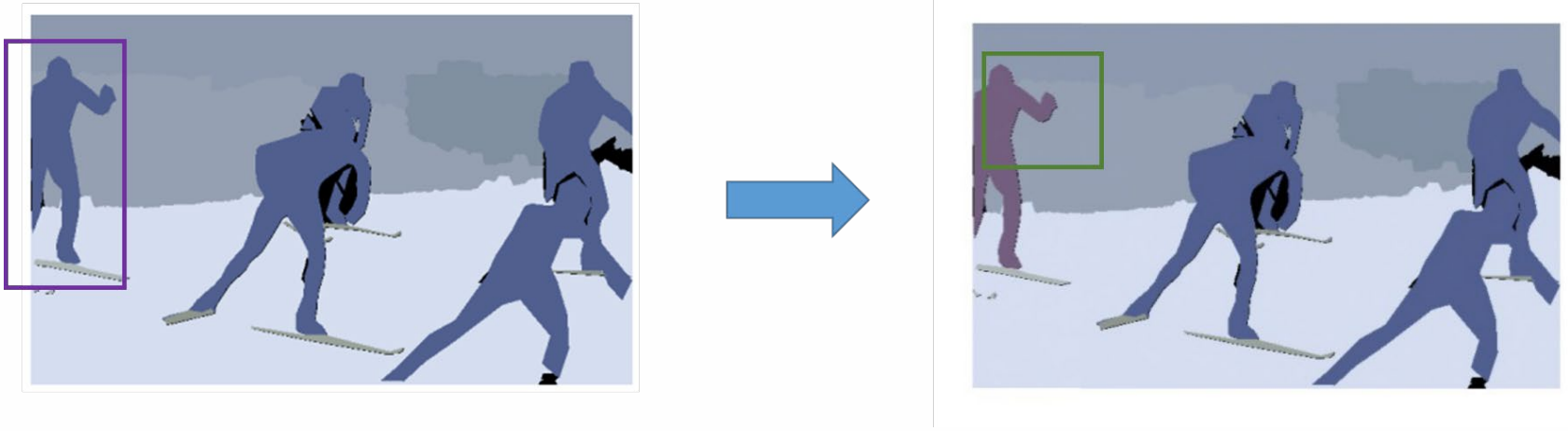
(b) Object detection.

(c) Semantic segmentation.

(d) Panoptic segmentation.

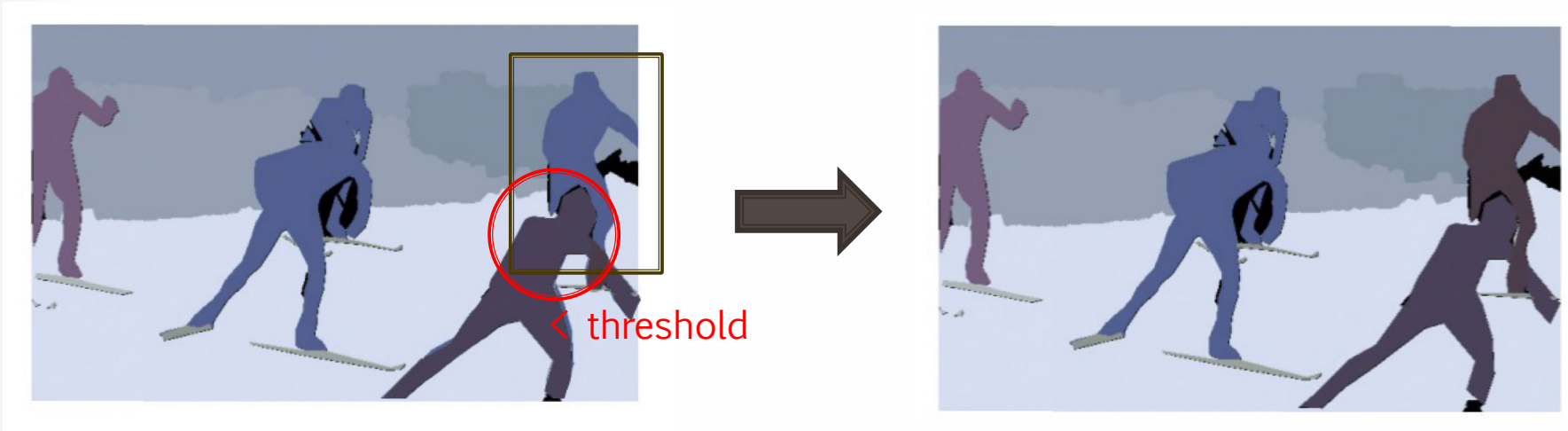
Panoptic Head

- iteratively “paste” each box to the semantic segmentation prediction by confidence from high to low
- “occupy” the pixels and hide false positive area from low-confident predictions.



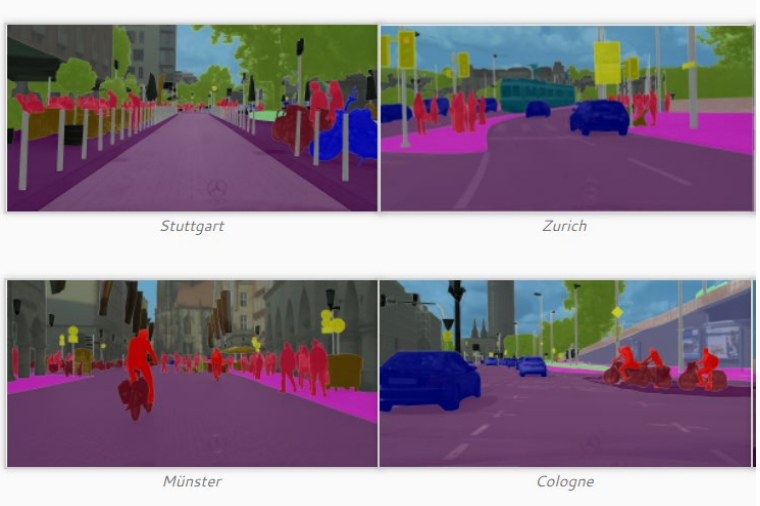
Panoptic Head

- If the intersection between the current mask and those already existing is larger than a threshold , discard this mask



Dataset

- COCO
- Cityscapes
- Mapillary Vistas



Training method

- 3 sub-tasks(MTL):
  - 1. bounding box localization
  - 2. border-ness regression
  - 3. semantic segmentation
- stochastic gradient descent (SGD):
  - batch size : 16
  - Weight decay : 0.0001
  - Momentum : 0.9
  - Step learning rate decay
    - initial learning rate : 0.01
    - decay rate : 0.1
  - Loss weights of semantic head are 0.7 and 1.0 on COCO and Cityscapes

$$L_{MTL} = \sum_i w_i \cdot L_i$$

COCO

- panoptic quality (PQ)

Table 1. **Panoptic segmentation results on COCO val.** The number in bracket stands for the length of shorter size of our train/test images. Other methods are trained/tested with images with shorter size length as 800. Superscripts Th and St stand for thing and stuff. '-' means inapplicable. We note that the running time includes the post-processing.

Models	PQ	SQ	RQ	PQ^{Th}	PQ^{St}	box mAP	mask mAP	mIoU	time	memory	FLOPS
JSIS-Net [3]	26.9	72.4	35.7	29.3	23.3	-	-	-	-	-	-
Panoptic FPN [10]	33.3	-	-	45.9	28.7	-	-	41.0	-	-	-
OANet [20]	39.0	77.1	-	43.5	24.9	-	-	-	-	-	-
SSAP [6]	36.5	**80.7**	44.8	40.1	32.0	-	-	-	-	-	-
Axial-DeepLab-L [32]	**43.9**	-	-	**48.6**	**36.8**	-	-	-	-	-	-
UPSNet [34]	42.3	78.0	52.4	48.5	33.4	37.8	**34.3**	54.3	202	10.4G	259G
LPSNet (800)	39.1	75.2	51.2	43.9	30.1	39.2	26.8	54.5	108	4.4G	139G
LPSNet (882)	41.9	77.2	51.5	46.3	32.5	**39.4**	29.6	56.1	127	4.9G	153G
LPSNet (964)	**42.4**	79.2	**52.7**	**48.0**	35.8	**39.5**	31.5	**59.8**	146	5.5G	167G

$$\text{PQ} = \underbrace{\frac{\sum_{(p,g)\in TP}\text{IoU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}$$

COCO

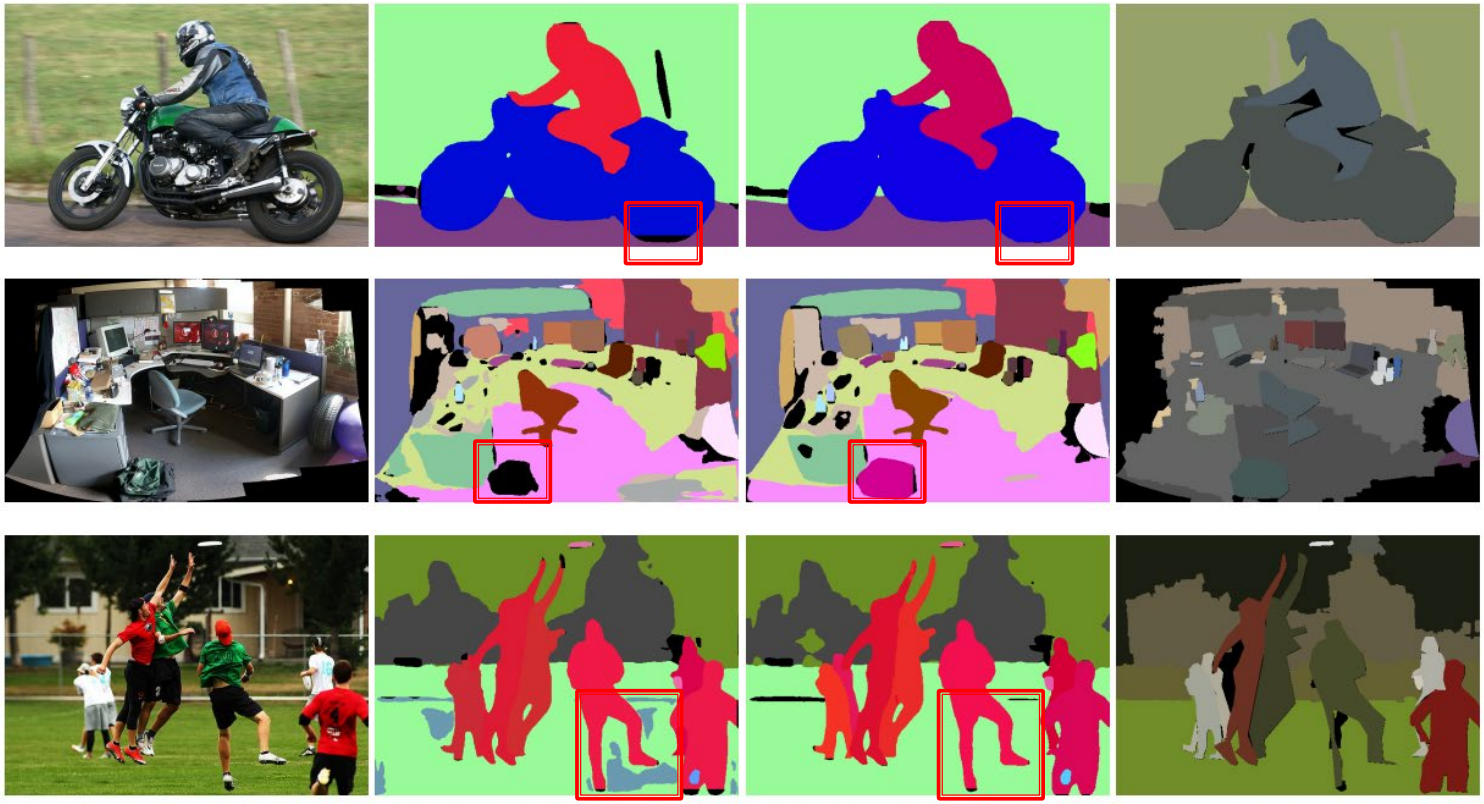


Figure 6. **Visual examples of panoptic segmentation on COCO dataset.** From left to right are input image, UPSNet output, LPSNet output, ground-truth.

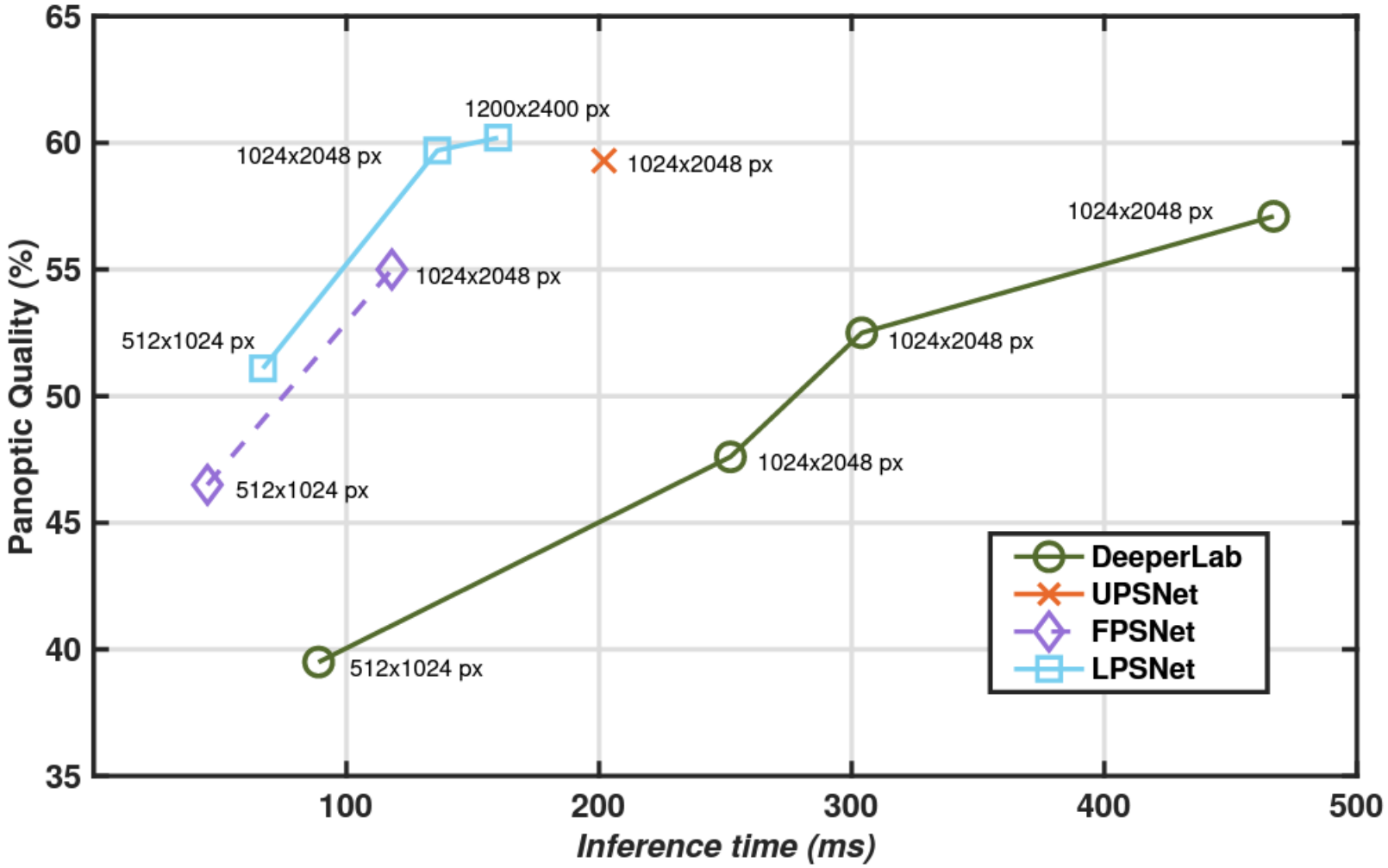
Cityscapes

- It is better performance of the LPSNet since many background existing in the Cityscapes dataset

Table 3. **Panoptic segmentation results on Cityscapes val.** Superscripts Th and St stand for thing and stuff. '-' means inapplicable.

Models	PQ	SQ	RQ	PQ^{Th}	PQ^{St}	box mAP	mask mAP	mIoU
Li *et al.* [15]	53.8	-	-	42.5	62.1	-	28.6	71.6
Panoptic FPN [10]	58.0	79.2	71.8	52.3	62.2	-	32.8	75.2
TASCNet [14]	55.9	-	-	50.5	59.8	-	-	-
SSAP [6]	58.4	-	-	50.6	-	-	**34.4**	-
UPSNet [34]	59.3	79.7	73.0	**54.6**	62.7	36.8	33.3	75.2
LPSNet (1024)	**59.7**	**79.9**	**73.6**	54.0	63.9	**38.4**	32.8	78.1
LPSNet (1200)	**60.4**	**80.3**	**74.0**	54.2	**64.5**	**38.5**	33.0	**78.6**
Multi-scale	PQ	SQ	RQ	PQ^{Th}	PQ^{St}	box mAP	mask mAP	mIoU
Panoptic FPN [10]	**61.2**	80.9	74.4	54.0	**66.4**	-	**36.4**	80.9
UPSNet [34]	60.1	80.3	73.5	**55.0**	63.7	37.1	33.3	76.8
LPSNet (1024)	60.5	80.7	73.5	53.7	64.8	**38.8**	33.2	80.8
LPSNet (1200)	**61.3**	**81.2**	**74.8**	54.5	65.6	**38.9**	33.6	**81.4**

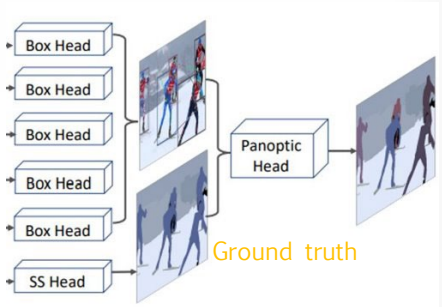
Cityscapes

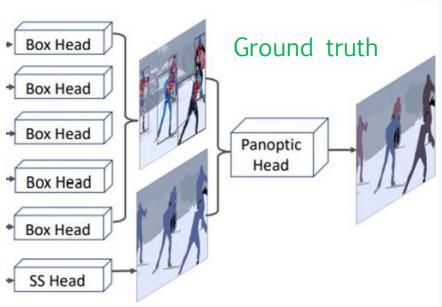


Mask head matters

Table 6. **With/without groundtruth semantic segmentation (SS) and bound boxes.** The experiments are conducted on COCO val set with ResNet-50 as backbone.

	box mAP	segmentation IoU	mask mAP	PQ
UPSNet	37.8	54.3	34.3	42.5
UPSNet + SS	37.8	-	46.1	72.0
UPSNet + Box	-	54.5	51.0	60.8
Ours	39.2	54.5	26.8	39.1
Ours + SS	39.2	-	47.0	74.1
Ours + Box	-	54.5	51.8	53.0





Conclusion

- LPSNet is a one-stage, anchor-free and proposal-free network
- LPSNet is more accurate and efficiency than popular two-stage approaches(UPSNet)
- Improve the mask head

CURRENT SOTA: INTERNIMAGE

CVPR 23

New SOTA: Yes, CNN is back…

- Title: InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions
- Authors: Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, Xiaogang Wang, Yu Qiao
- URL: https://arxiv.org/pdf/2211.05778.pdf
- Code: https://github.com/opengvlab/internimage

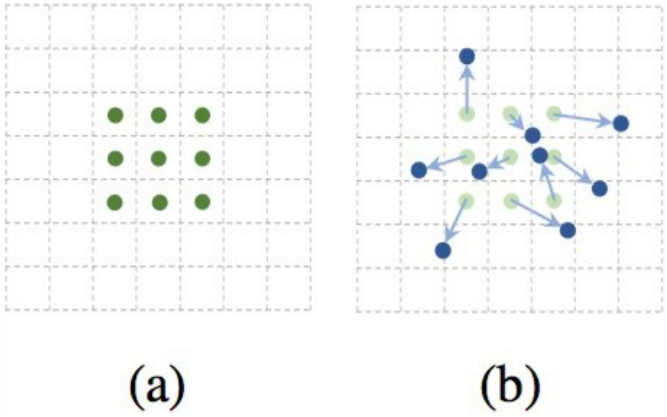
InternImage Features

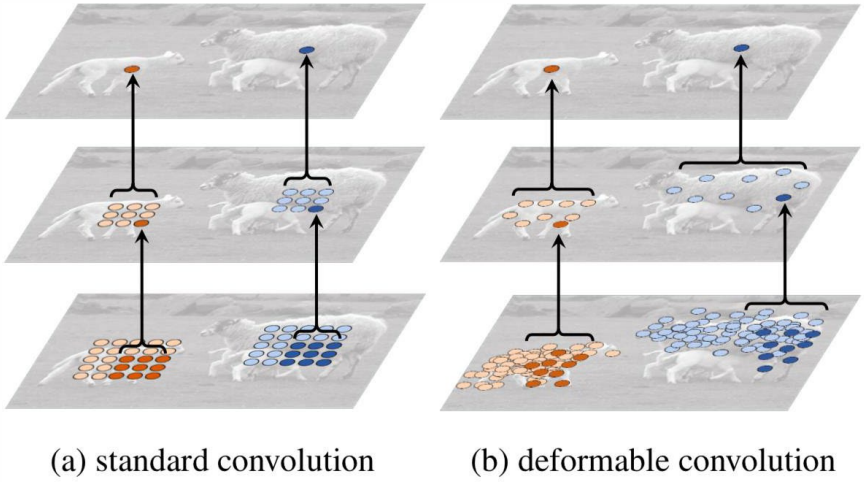
- Large-scale CNN-based backbone model, InternImage,
  - Over 1 billion parameters and 400 million data points.
  - By improving the deformable convolution DCN
- InternImage has been compared to state-of-the-art CNNs and large-scale ViTs (Vision Transformers) in various tasks,
- For object detection tasks (COCO dataset), using InternImage as the backbone and DINO as the detector, achieves SOTA results
- For semantic segmentation tasks (ADE20K), we also achieved the highest accuracy.

Deformable convolution

- Traditional convolution has limitations in capturing long-range dependencies and adaptive spatial aggregation.

- Deformable convolution overcomes these limitations by introducing offset-based adaptive spatial sampling.





DCN for CNN

- Conduct a regular convolution on the input feature map.
- Obtain information about the offset for each pixel.
- Return the offset information to the filter.
- Sample from the input feature map based on the offset.
- Deformable convolution

- Convolutional layer outputs offset field with $2N$ channels.
- N represents the number of filter elements.
- Each element requires changes in both dx and dy, hence the doubling of channels.

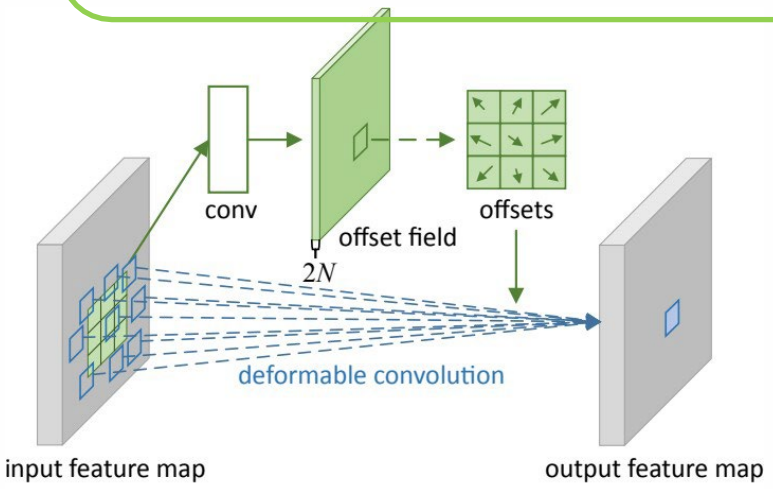
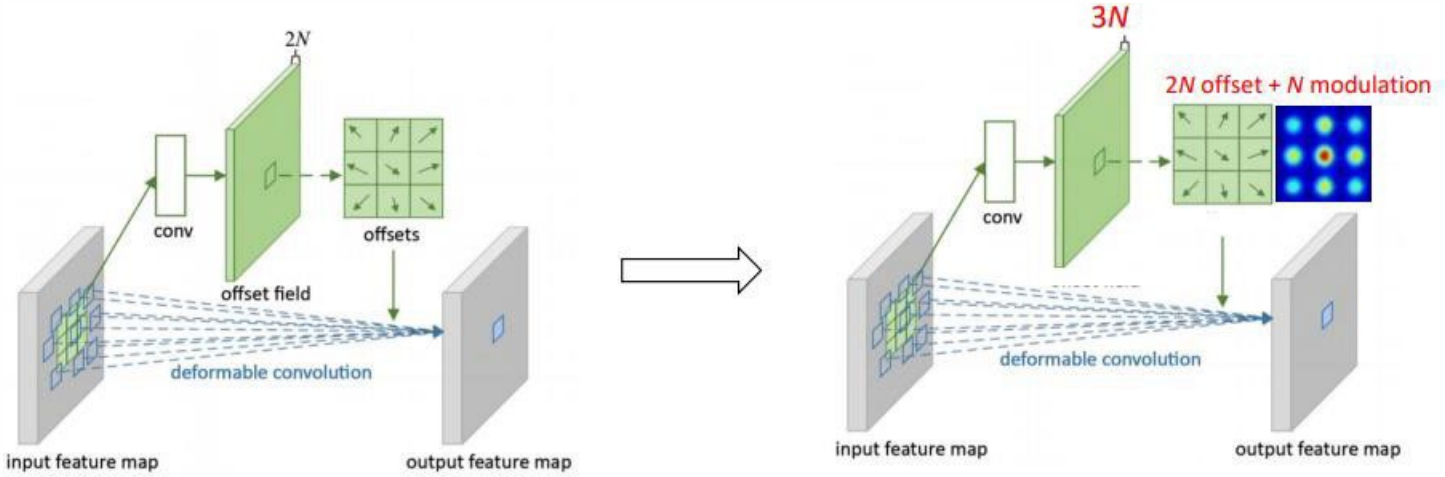


Figure 2: Illustration of 3×3 deformable convolution.

DCN? First one?

- No, actually, DCN was proposed in 2018 early
- DCNv2 was proposed in 2018 late



$$y(p_0) = \sum_{n=0}^{N} w(p_n) \cdot x(p_0 + p_n + \Delta p_n) \cdot \Delta m_n$$

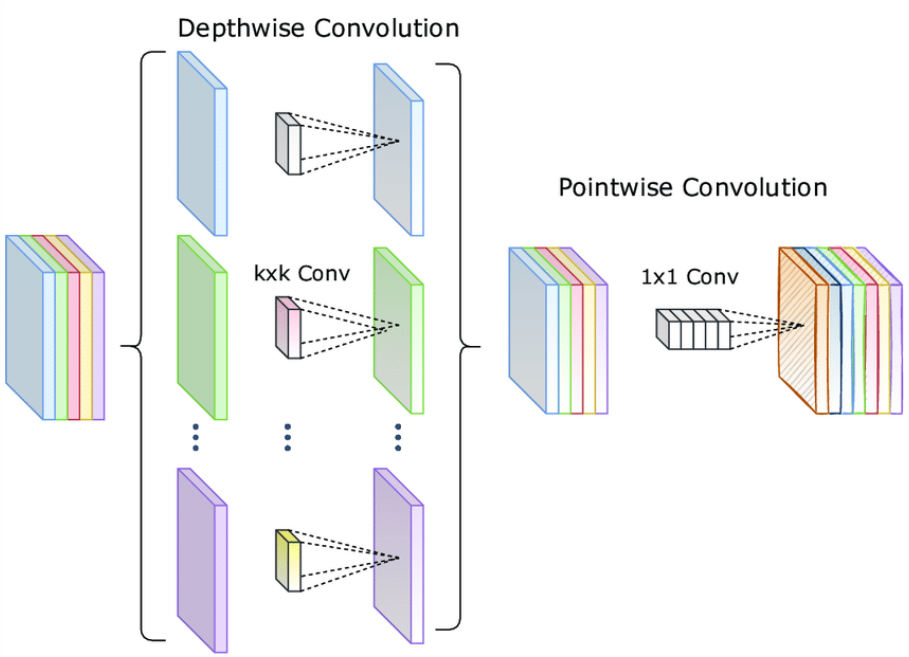
Δmn ： [0,1] Adding a modulation scalar that takes values in the range [0, 1]. By multiplying each pixel with a different scalar value, it increases flexibility and improves performance.

DCNv3 (This paper)

- Based on DCNv2
  - Separable convolution is incorporated in DCNv2.
    - This technique enhances efficiency and reduces computational requirements.
    - It allows for more flexible and resource-efficient convolution operations.
  - The multi-group mechanism is introduced.
    - This mechanism improves the parallelism and scalability of the model.
    - It enables better utilization of hardware resources and faster processing.
  - Normalization of the modulation scalar is performed along the samples.
    - This normalization ensures consistent and stable scaling of the modulated features.
    - It enhances the overall performance and accuracy of the model.

Accelerating the convolution

- Sparable convolution is introduced.
- Regular convolution is separated into spatial and channel directions.
- Spatial convolution maintains the output map size.
- Channel convolution is performed using a 1x1 filter for dimension compression.
- This approach greatly reduces parameters.



DCNv3 (This paper)

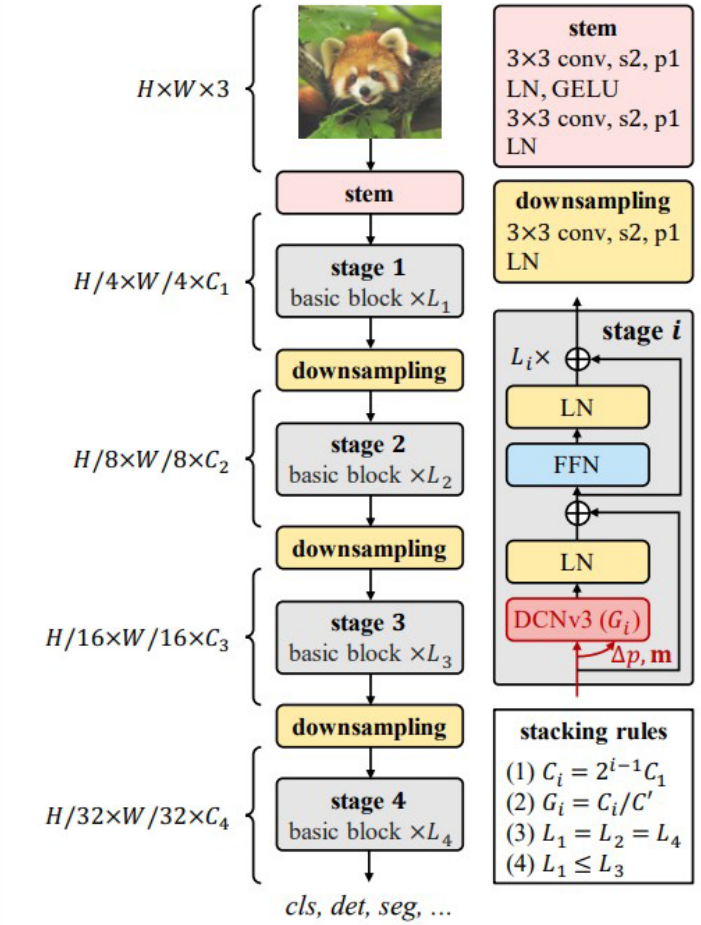
- Introducing the Multi-group mechanism.
- Inspired by Multi-head Self-Attention.
- Grouping channels to acquire more complex representations.
- Normalization of modulation scalar along the samples.

$$\boldsymbol{y}(\boldsymbol{p_0}) = \sum_{g=0}^{G} \sum_{n=0}^{N} \boldsymbol{w}_g(\boldsymbol{p_n}) \cdot \boldsymbol{x}(\boldsymbol{p_0} + \boldsymbol{p_n} + \Delta \boldsymbol{p}_{gn}) \cdot \Delta m_{gn}$$

- Issue:
  - The sum of weights within the filters varies greatly from 0 to *K*, leading to unstable learning.
  - ⇒ Adopting softmax function to ensure the sum of weights within each filter is 1. This stabilizes the learning process.

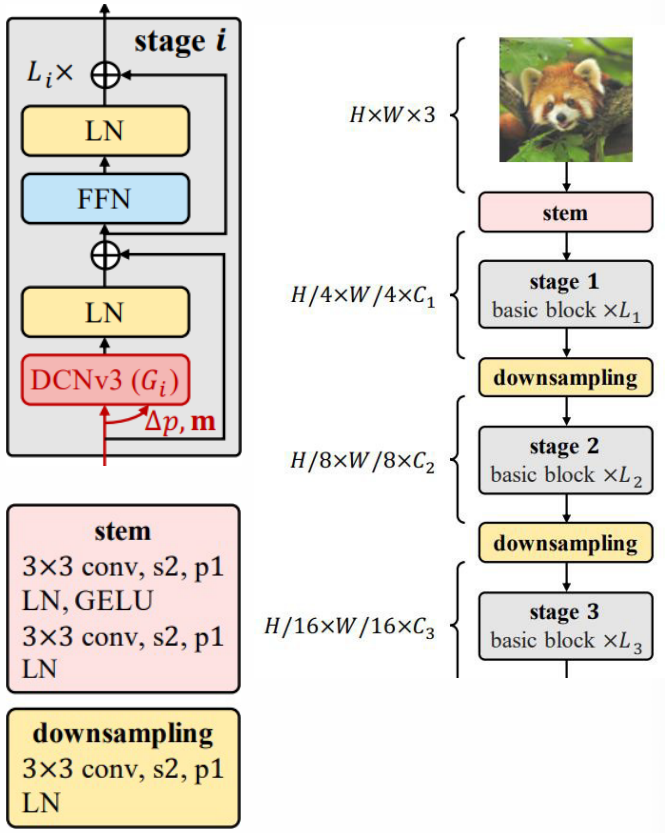
Architecture of InternImage

- Multi-group mechanism enhances representation.
- Inspired by Multi-head Self-Attention.
- Grouping channels for complex representations.
- Normalizing modulation scalar for stability.
- Problem: Weight sum instability.
- Solution: Softmax function for stable learning.



Basic Block Design

- Basic block
  - It mimics the structure of ViT.
  - Uses DCN (Dynamic Convolutional Networks).
  - Applies Layer Normalization.
  - Utilizes Feed-Forward Networks.
  - Implements
- Stem and Downsampling
  - Changes the scale of feature maps.
  - Stem reduces the input image resolution by 1/4.
  - Downsampling decreases the resolution of feature maps by 1/2.



Optimal parameter settings for the origin model

- Staking rules:
  - The model consists of four stages, each with hyperparameters such as channel numbers, group numbers, and the number of basic blocks, resulting in a total of 12 parameters that determine the model. The search space is too large.
  - Based on previous research and design experience, three restrictions are imposed on the structure to search for the optimal model.

- $C_i = 2^{i-1}C_1$
- $G_i = C_i/C'$
- $L_1 = L_2 = L_4 \ (L_1 \leq L_3)$

1.i-th channel: Ci
2.DCNv3 groups: Ci / C'
3.$L1 \leq L3$: Number of basic blocks in each stage

Scaling up

- Scaling rules:
  - We use the stacking rule to determine the optimal origin model [C1 = 64, C' = 16, L1 = 4, L3 = 18]. Then, we apply the compound scaling method to scale the depth and channel numbers, increasing the model's size (see Appendix for details).

model name	C_1	C'	$L_{1,2,3,4}$	#params
InternImage-T (origin)	64	16	4, 4, 18, 4	30M
InternImage-S	80	16	4, 4, 21, 4	50M
InternImage-B	112	16	4, 4, 21, 4	97M
InternImage-L	160	16	5, 5, 22, 5	223M
InternImage-XL	192	16	5, 5, 24, 5	335M
InternImage-H	320	32	6, 6, 32, 6	1.08B

Results: Totally SOTA

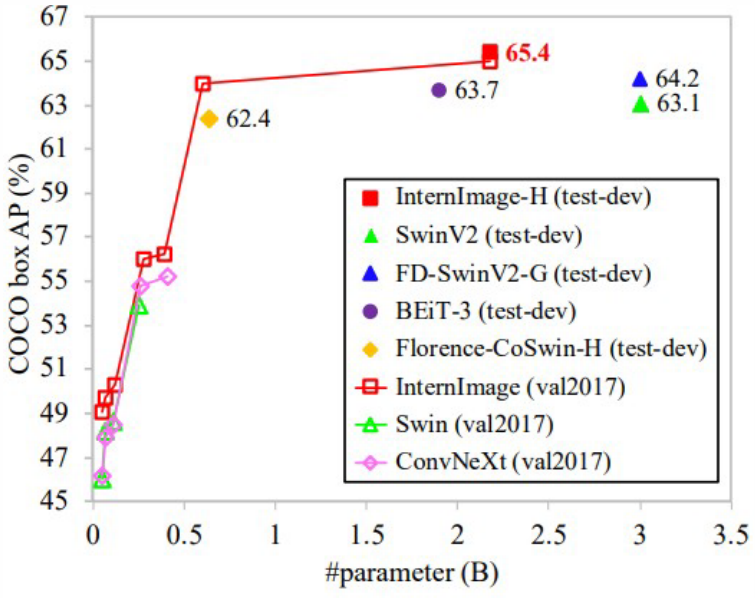


Figure 2. **Performance comparison on COCO of different backbones.** The proposed InternImage-H achieves a new record 65.4 box AP on COCO test-dev, significantly outperforming state-of-the-art CNNs and large-scale ViTs.

method	crop size	#params	#FLOPs	mIoU (SS)	mIoU (MS)
Swin-T [2]	512^2	60M	945G	44.5	45.8
ConvNeXt-T [21]	512^2	60M	939G	46.0	46.7
SLaK-T [29]	512^2	65M	936G	47.6	–
InternImage-T (ours)	512^2	59M	944G	47.9	48.1
Swin-S [2]	512^2	81M	1038G	47.6	49.5
ConvNeXt-S [21]	512^2	82M	1027G	48.7	49.6
SLaK-S [29]	512^2	91M	1028G	49.4	–
InternImage-S (ours)	512^2	80M	1017G	50.1	50.9
Swin-B [2]	512^2	121M	1188G	48.1	49.7
ConvNeXt-B [21]	512^2	122M	1170G	49.1	49.9
RepLKNet-31B [22]	512^2	112M	1170G	49.9	50.6
SLaK-B [29]	512^2	135M	1172G	50.2	–
InternImage-B (ours)	512^2	128M	1185G	50.8	51.3
Swin-L‡ [2]	640^2	234M	2468G	52.1	53.5
RepLKNet-31L‡ [22]	640^2	207M	2404G	52.4	52.7
ConvNeXt-L‡ [21]	640^2	235M	2458G	53.2	53.7
ConvNeXt-XL‡ [21]	640^2	391M	3335G	53.6	54.0
InternImage-L‡ (ours)	640^2	256M	2526G	53.9	54.1
InternImage-XL‡ (ours)	640^2	368M	3142G	55.0	55.3
SwinV2-G$^\#$ [16]	896^2	3.00B	–	–	59.9
InternImage-H$^\#$ (ours)	896^2	1.12B	3566G	59.9	60.3
BEiT-3$^\#$ [17]	896^2	1.90B	–	–	62.8
FD-SwinV2-G$^\#$ [26]	896^2	3.00B	–	–	61.4
InternImage-H$^\#$ (ours) + Mask2Former [80]	896^2	1.31B	4635G	62.5	62.9

Table 5. **Semantic segmentation performance on the ADE20K validation set.** The FLOPs are measured with 512×2048, 640×2560, or 896×896 inputs according to the crop size. "SS" and "MS" means single-scale and multi-scale testing, respectively.

Open Source Frameworks

- Lots of good implementations on GitHub!

- TensorFlow Detection API:
- https://github.com/tensorflow/models/tree/master/research/object_detection
  - Faster RCNN, SSD, RFCN, Mask R-CNN

- Caffe2 Detectron:
- https://github.com/facebookresearch/Detectron
  - Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, R-FCN

- Finetune on your own dataset with pre-trained models