Introduction to Computer Vision: Other tasks of visual perception

Natalia Neverova

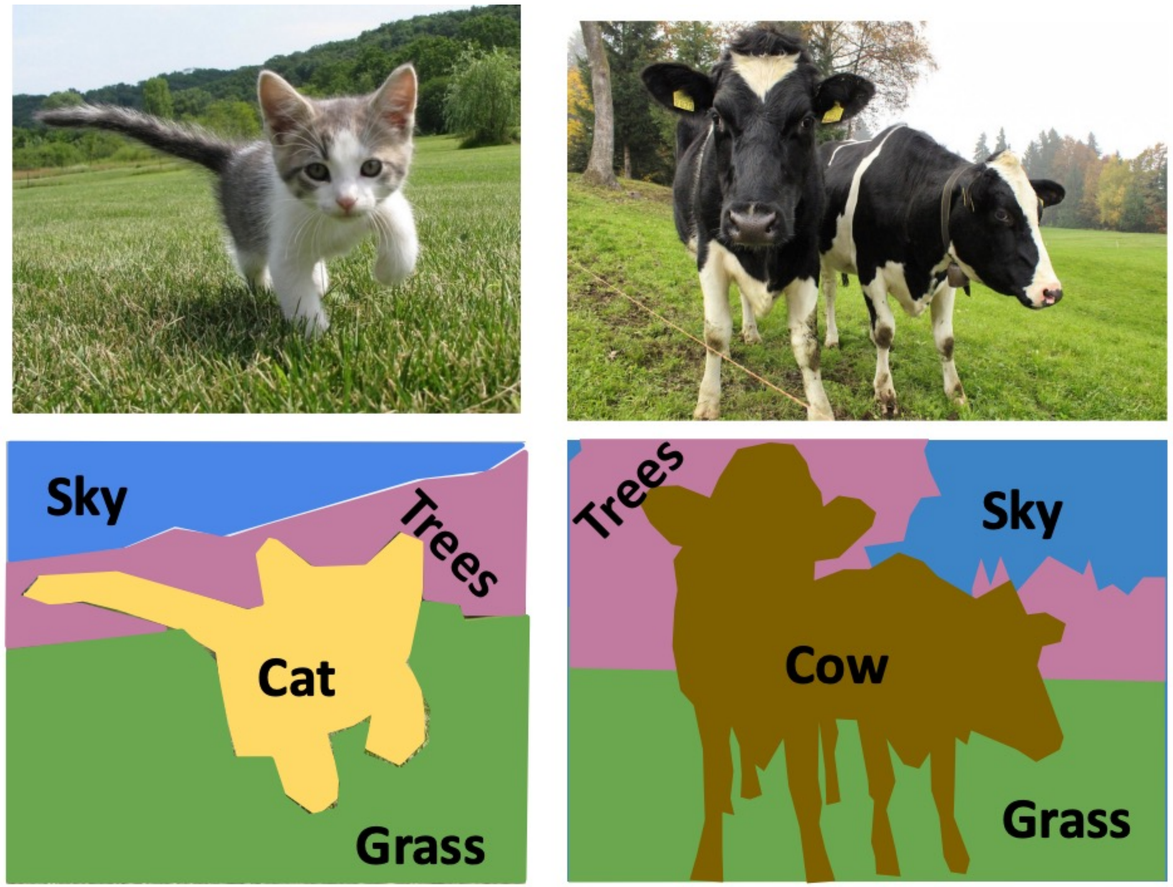
Last lecture's recap

- Principles of multi-object detection
- Region proposals
- Bounding box regression
- R-CNN training and inference
- Non-Max suppression
- Modern object detectors

Today's focus

- Semantic segmentation
- Extending R-CNN to other tasks
- Instance segmentation
- Panoptic segmentation
- Human pose estimation
- .. and others

Semantic segmentation

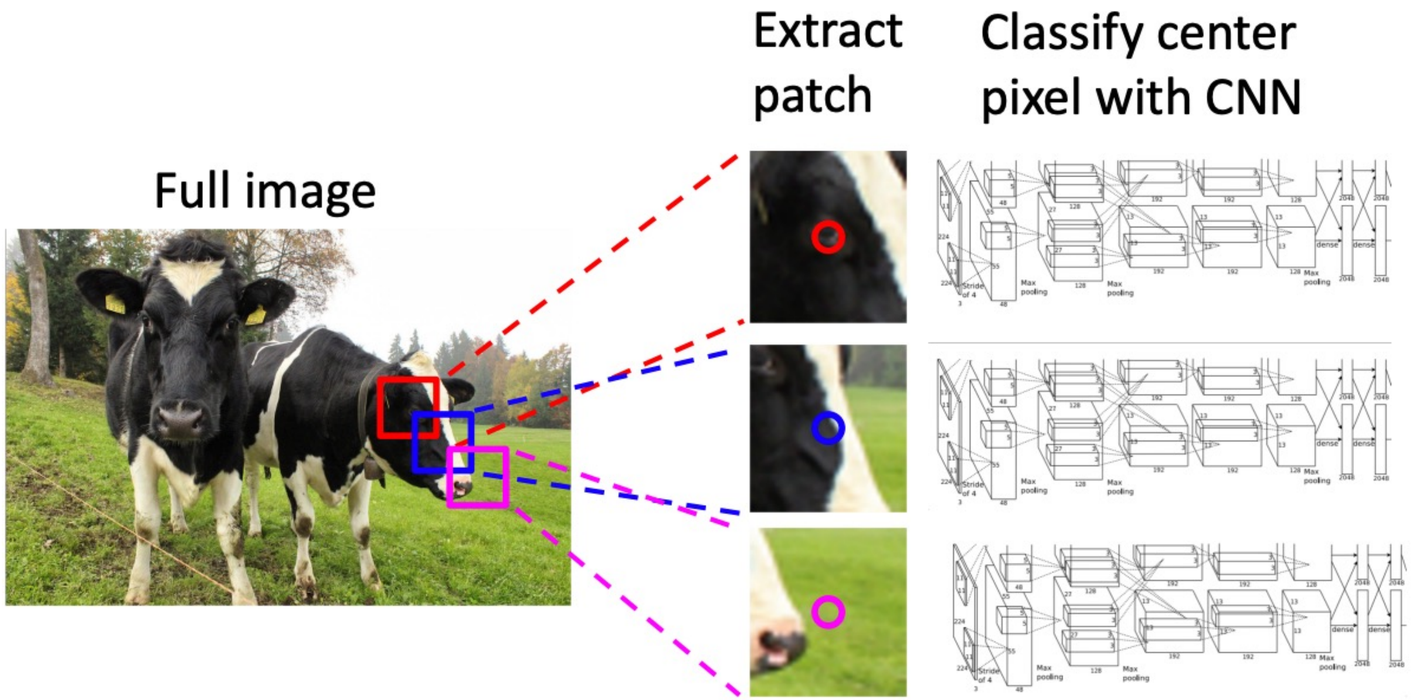


Forget about detection for a moment: there are no objects, just pixels!

Semantic segmentation task: label each pixel in the image with a category label (“cat”, “grass”, “sky”, ...)

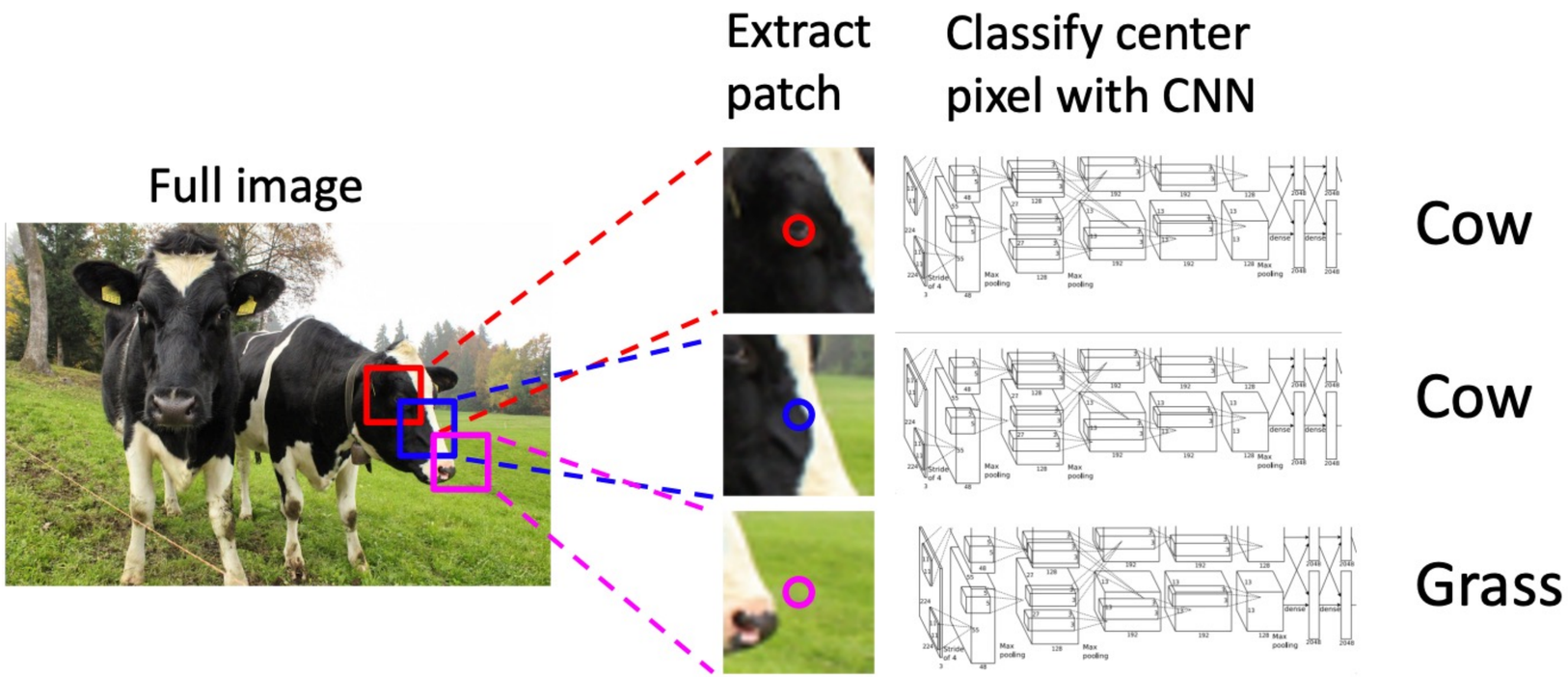
Don’t differentiate object instances, only care about pixels

Semantic segmentation



Idea similar to what we have seen before: **sliding window**, centered at every pixel

Semantic segmentation

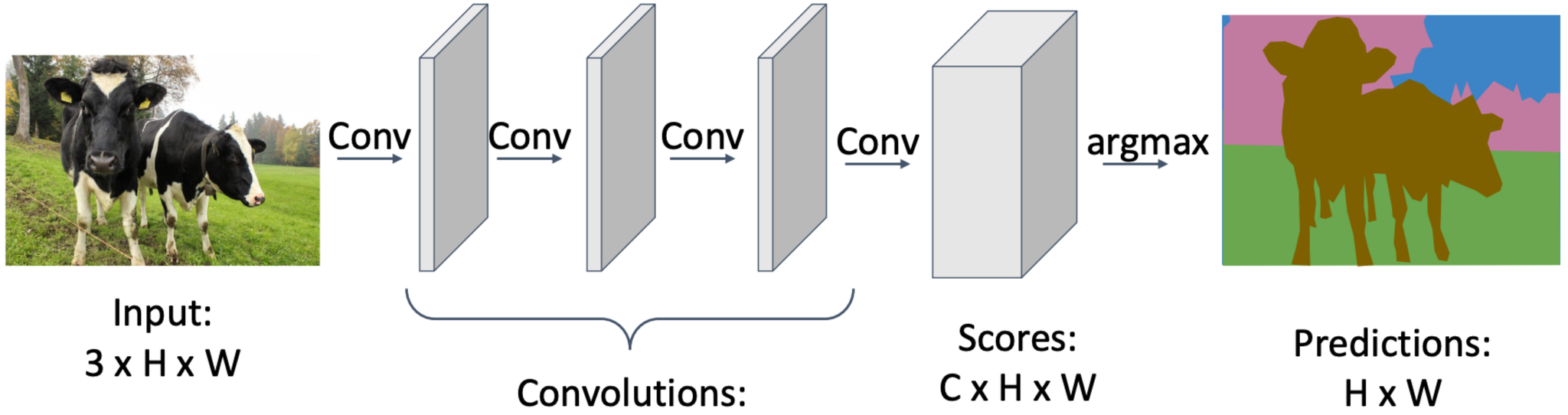


Idea similar to what we have seen before: **sliding window**, centered at every pixel

Same problem as before: it's very inefficient! Not reusing shared features between overlapping patches

Semantic segmentation

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

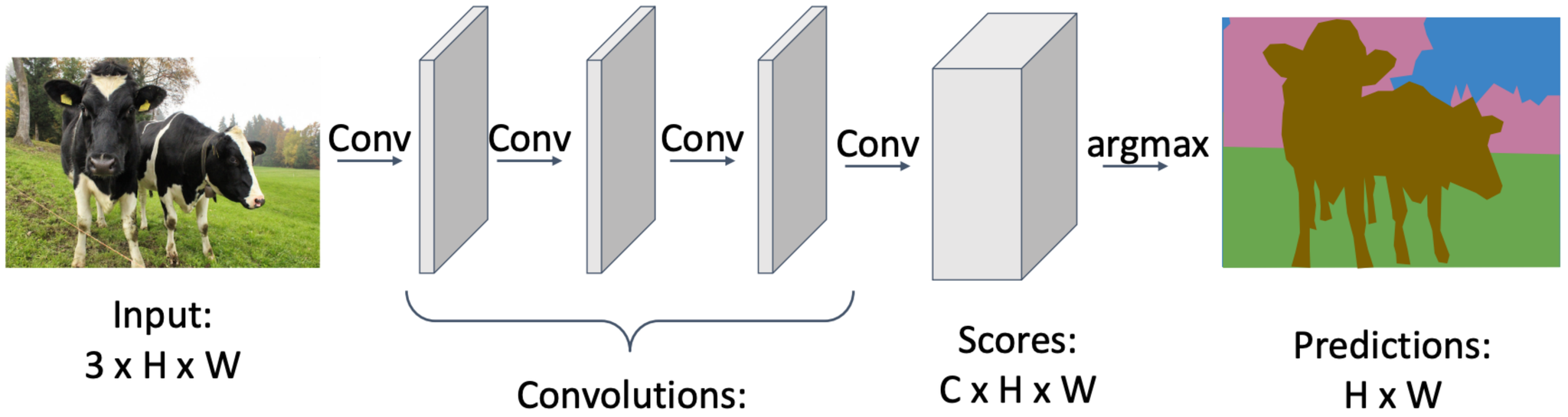


Fully Convolutional Network (FCN)

Loss function is per-pixel cross-entropy

Semantic segmentation: FCN

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

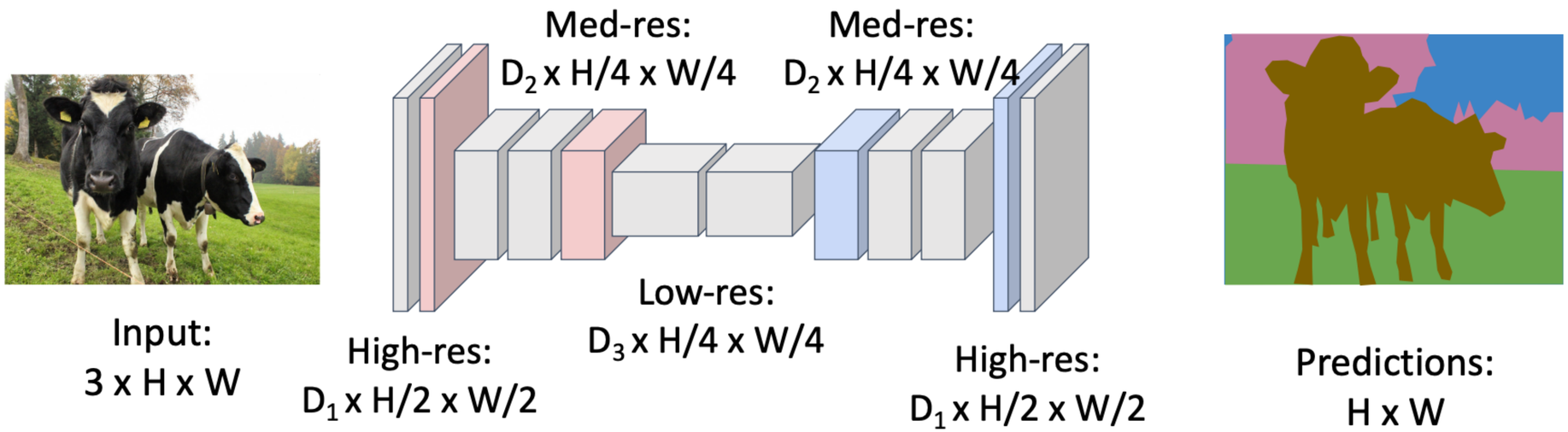


Problem #1: Effective receptive field size is linear in number of conv layers: receptive field is 1+2L for L 3x3 conv

Problem #2: Convolution on high-res images is expensive!

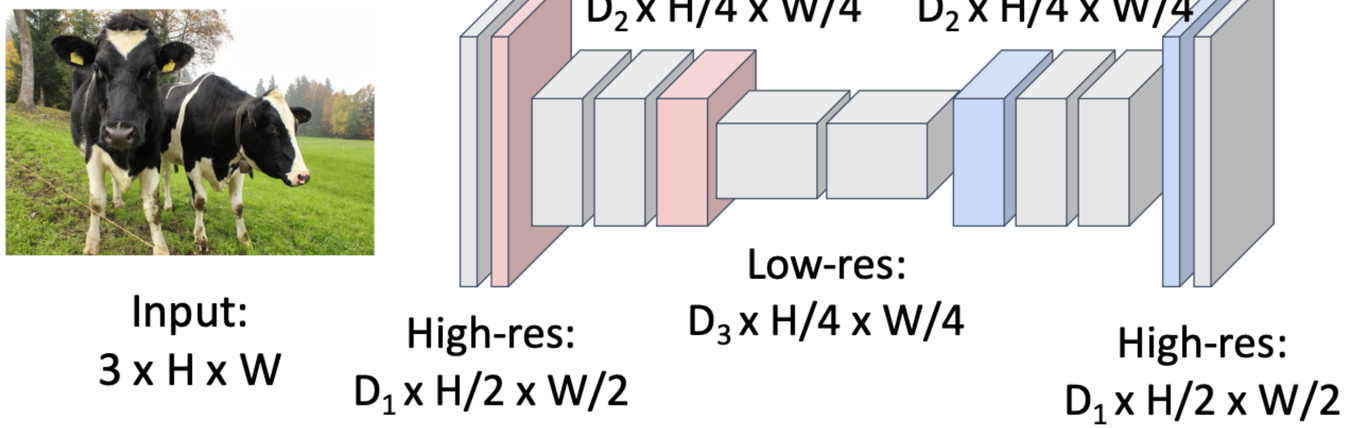
Semantic segmentation: FCN

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



Semantic segmentation: FCN

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

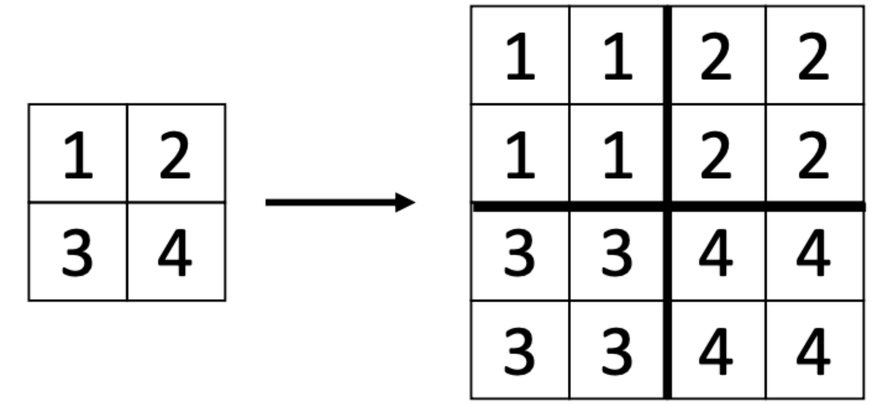


Predictions:
H x W

Downsampling:
pooling, strided convolution

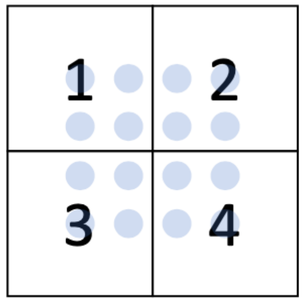
Upsampling:
interpolation, transposed convolution

In-network upsampling: nearest neighbor



Copy values from neighboring cells

In-network upsampling: bilinear interpolation



1.00	1.25	1.75	2.00
1.50	1.75	2.25	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

Input: C x 2 x 2

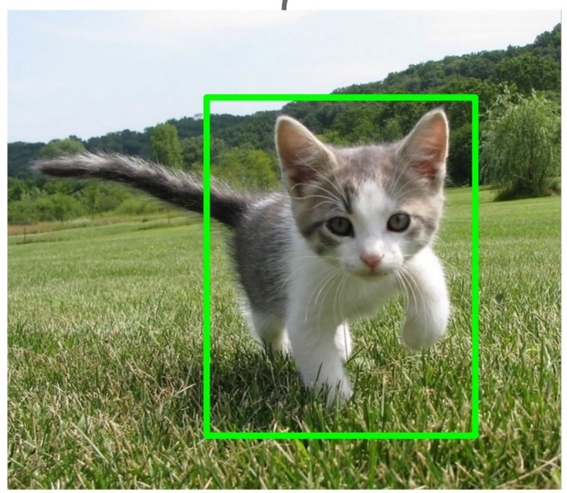
Output: C x 4 x 4

$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|) \quad i \in \{\lfloor x \rfloor - 1, \ldots, \lceil x \rceil + 1\} \quad j \in \{\lfloor y \rfloor - 1, \ldots, \lceil y \rceil + 1\}$$

Use two closest neighbors in x and y to construct linear approximations

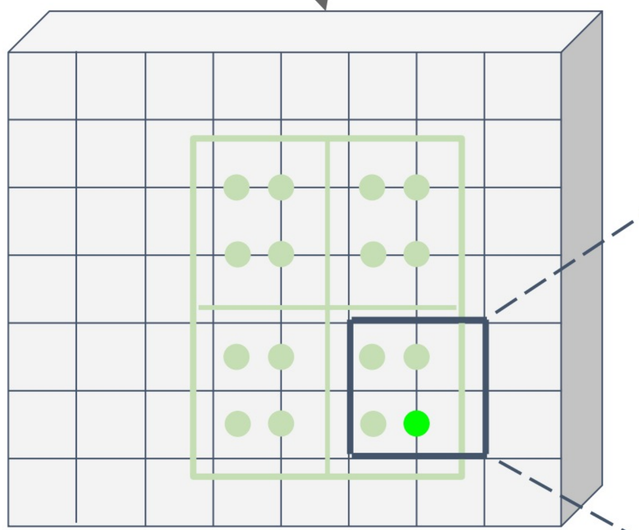
same algorithm as in ROI Align

Reminder from last lecture: cropping features – ROI Align

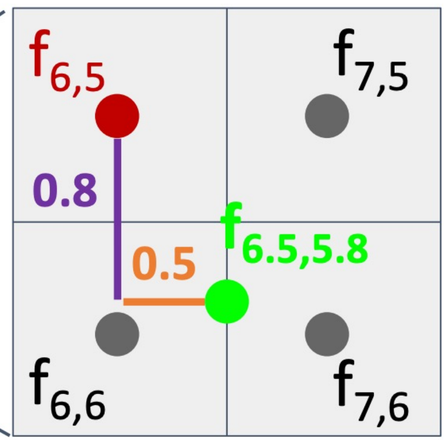


Project proposal onto features

No "snapping"



sample features at regularly-spaced points in each subregion using **bilinear interpolation**

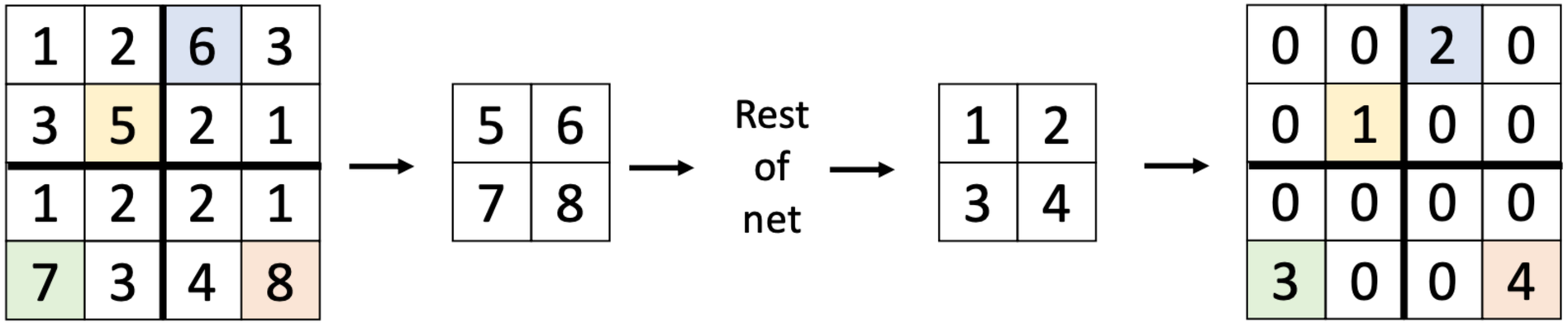


$$f_{xy} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_i|)$$

$f_{6.5,5.8} = (f_{6,5} * 0.5 * 0.2) + (f_{7,5} * 0.5 * 0.2)$
$+ (f_{6,6} * 0.5 * 0.8) + (f_{7,6} * 0.5 * 0.8)$

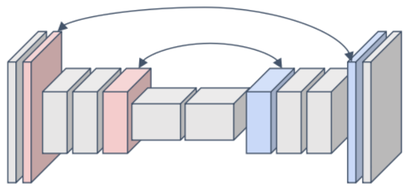
Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells:

In-network upsampling: “max unpooling”



Max Pooling: Remember which position had the max

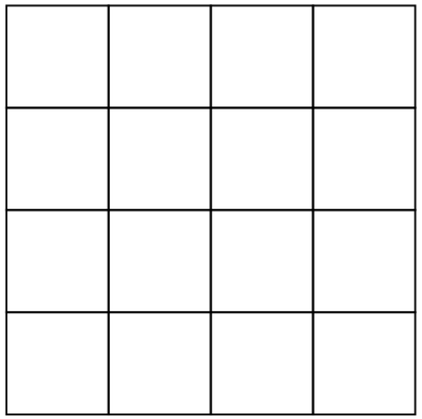
Max Pooling: Place into remembered positions



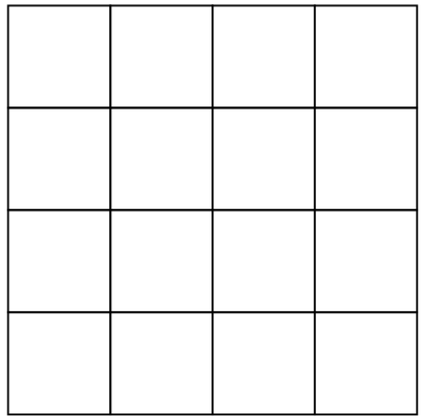
Pair each downsampling layer with an upsampling layer

Learnable upsampling: deconvolution

Recall: Normal 3x3 convolution, stride 1



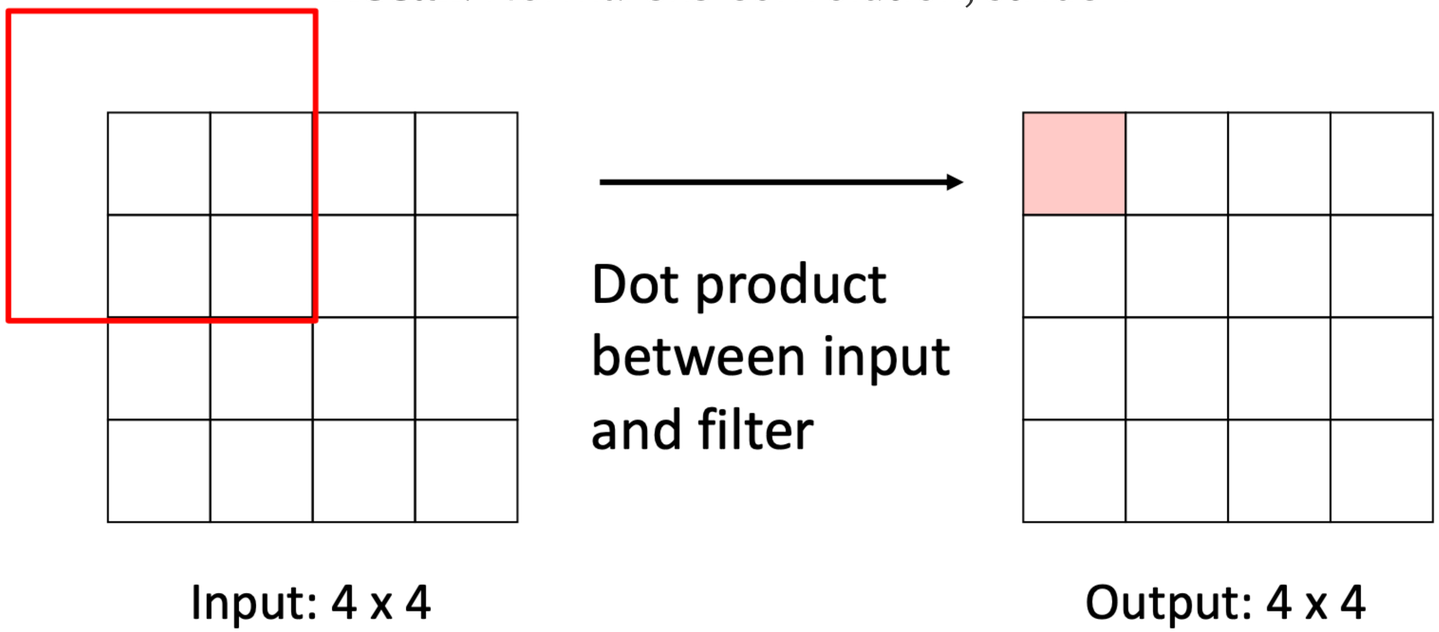
Input: 4 x 4



Output: 4 x 4

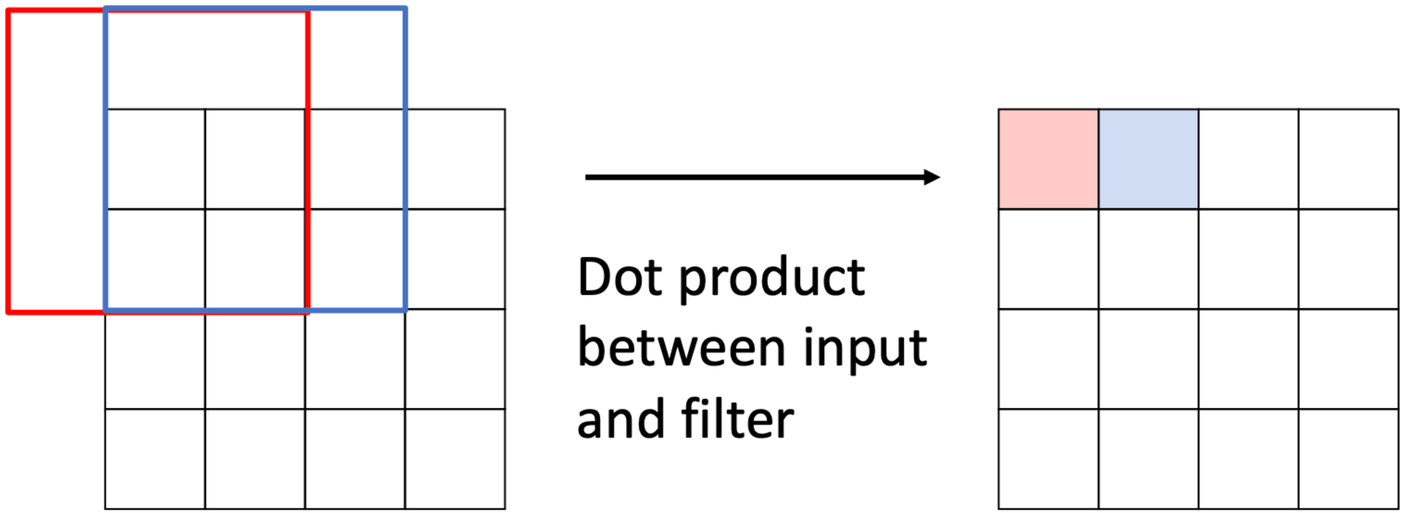
Learnable upsampling: deconvolution

Recall: Normal 3x3 convolution, stride 1



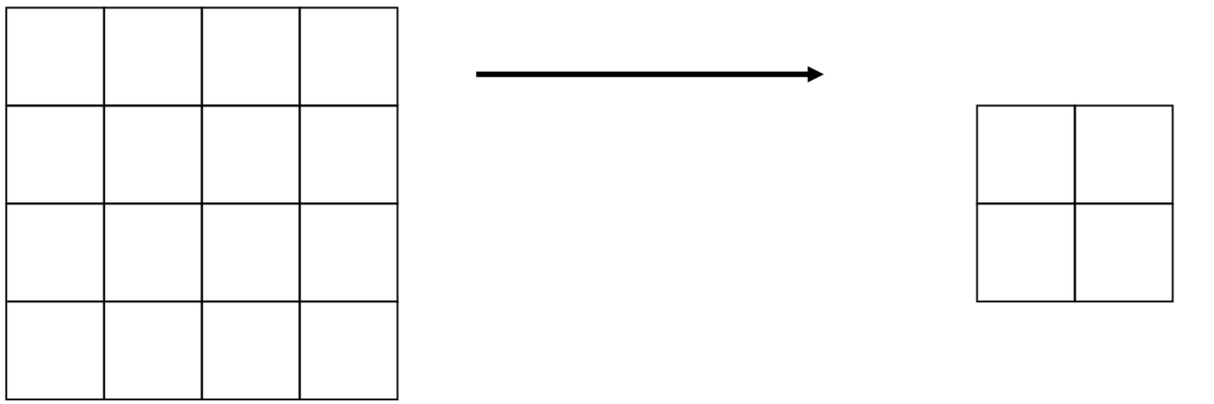
Learnable upsampling: deconvolution

Recall: Normal 3x3 convolution, stride 1



Learnable upsampling: deconvolution

Recall: Normal 3x3 convolution, stride 2

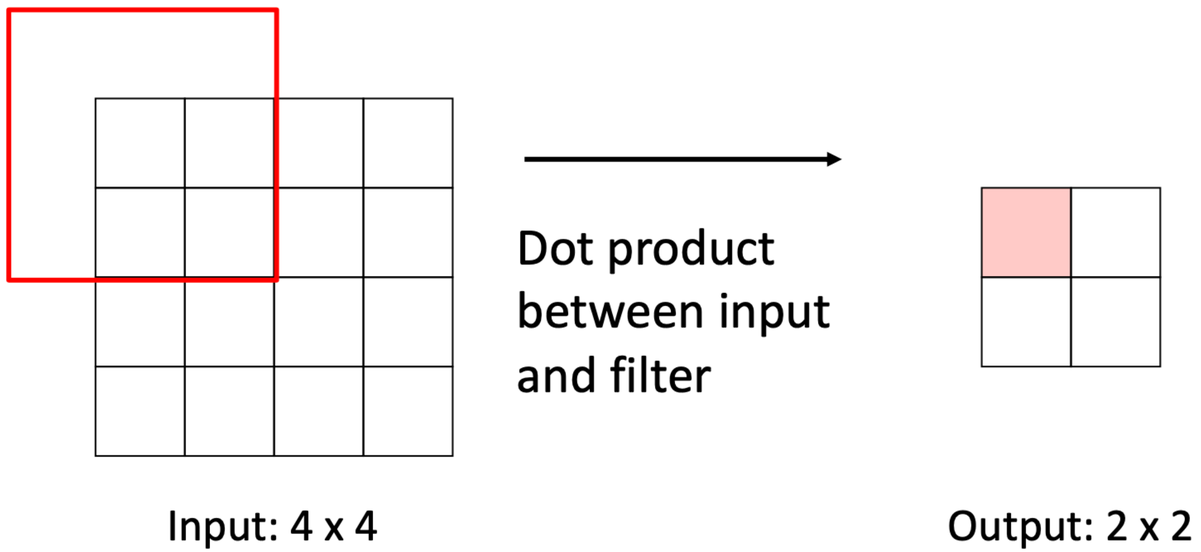


Input: 4 x 4

Output: 2 x 2

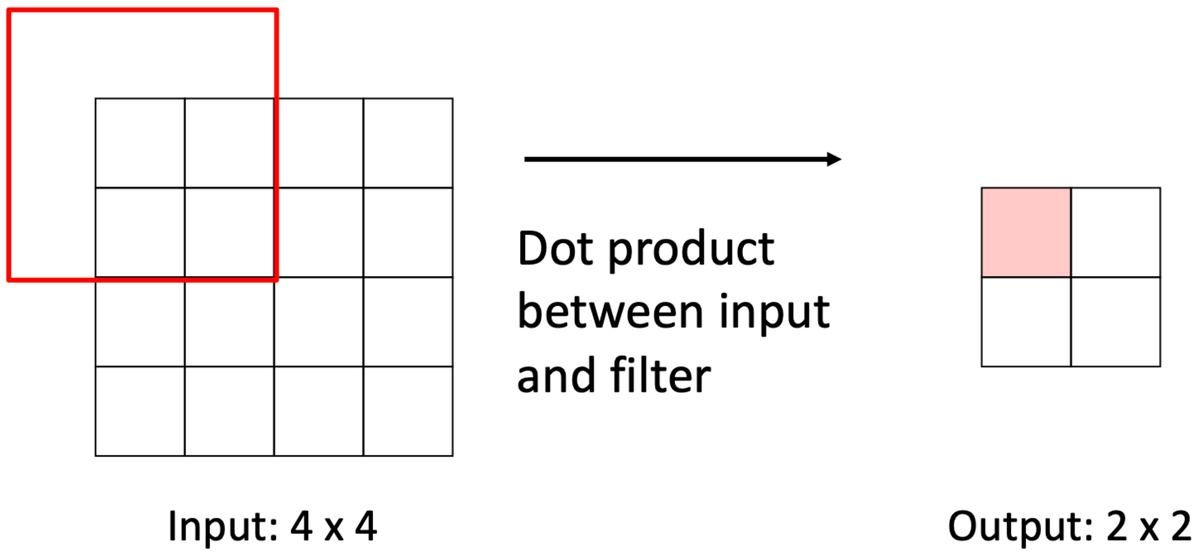
Learnable upsampling: deconvolution

Recall: Normal 3x3 convolution, stride 2



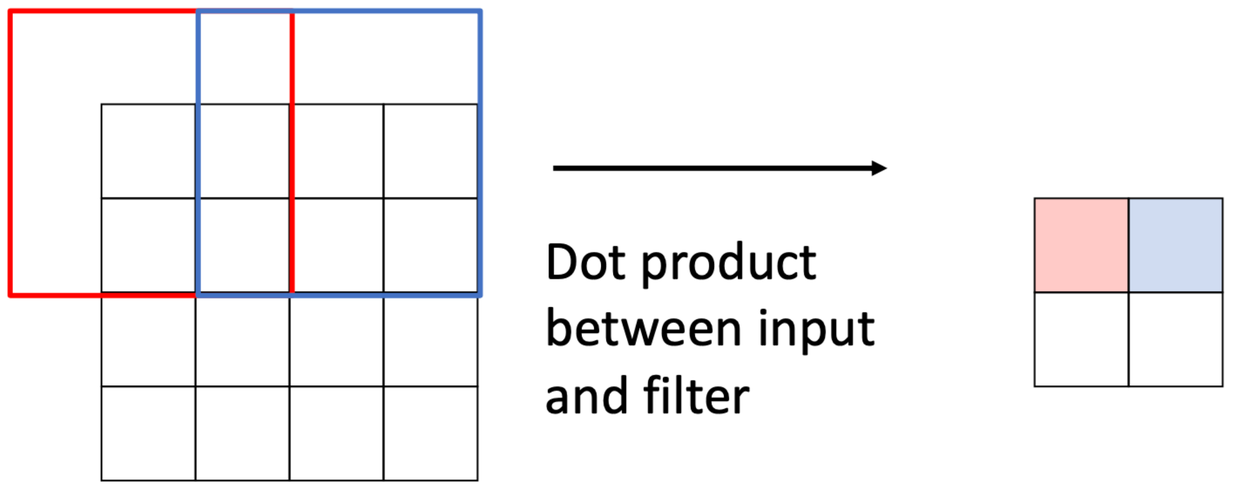
Learnable upsampling: deconvolution

Recall: Normal 3x3 convolution, stride 2



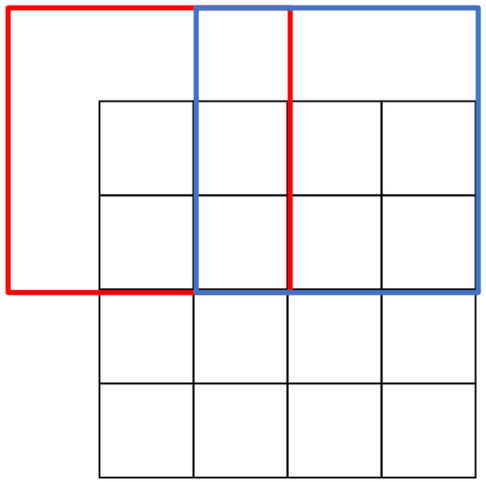
Learnable upsampling: deconvolution

Recall: Normal 3x3 convolution, stride 2



Learnable upsampling: deconvolution

Recall: Normal 3x3 convolution, stride 2

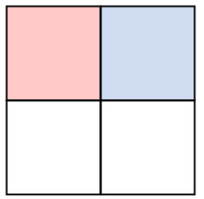


Input: 4 x 4

Convolution with stride > 1 is “learnable downsampling”

Can we use stride < 1 for “learnable upsampling”?

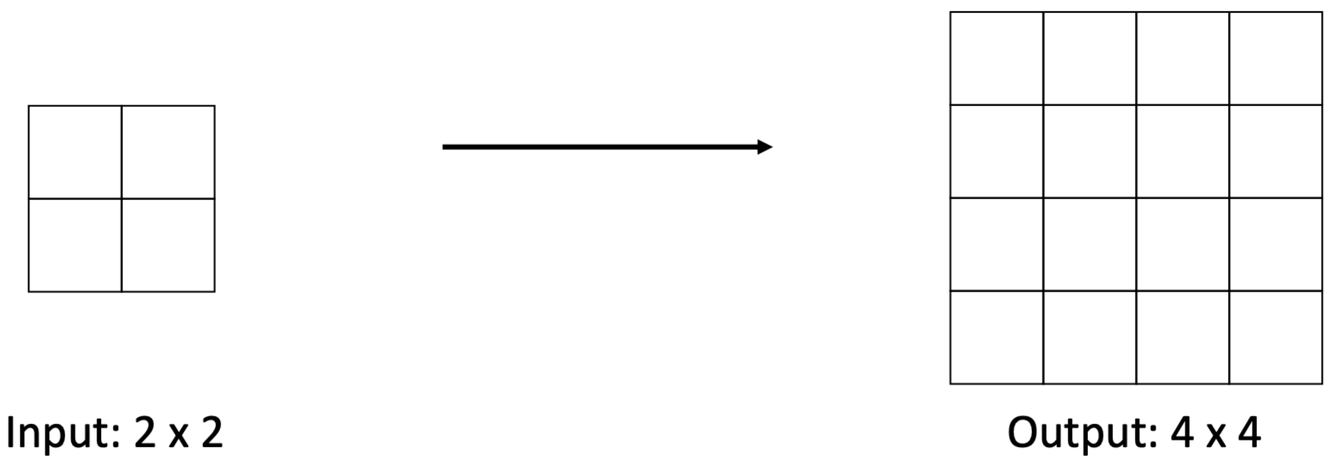
Dot product between input and filter



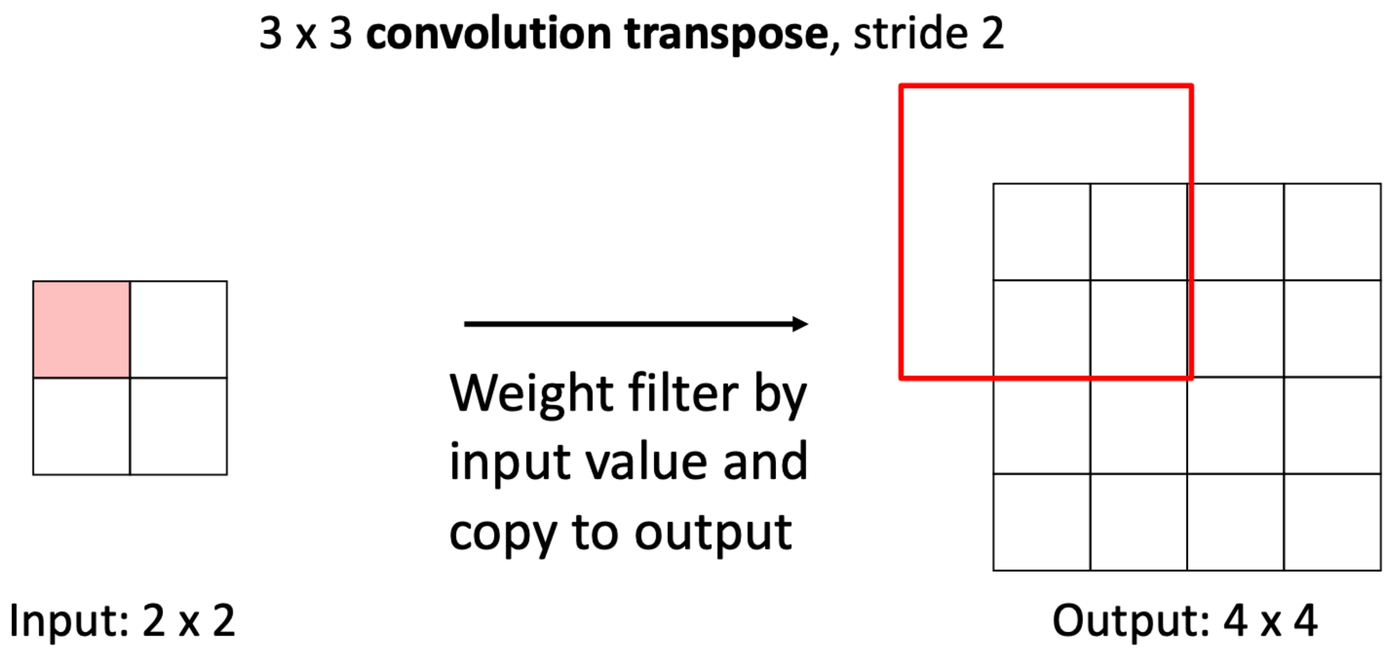
Output: 2 x 2

Learnable upsampling: deconvolution

3x3 **convolution transpose**, stride 2



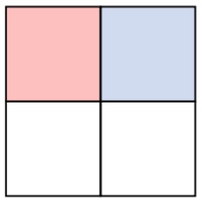
Learnable upsampling: deconvolution



Learnable upsampling: deconvolution

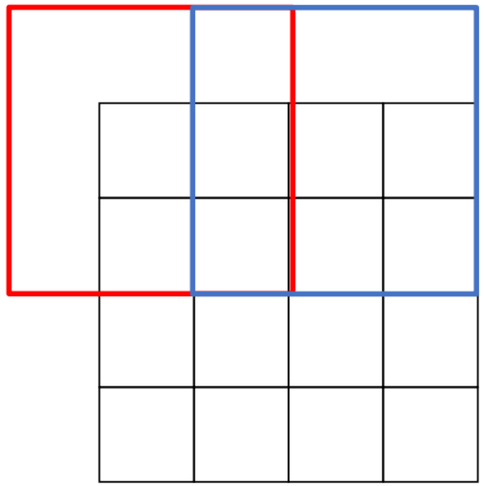
3 x 3 **convolution transpose**, stride 2

Filter moves 2 pixels in output
for every 1 pixel in input



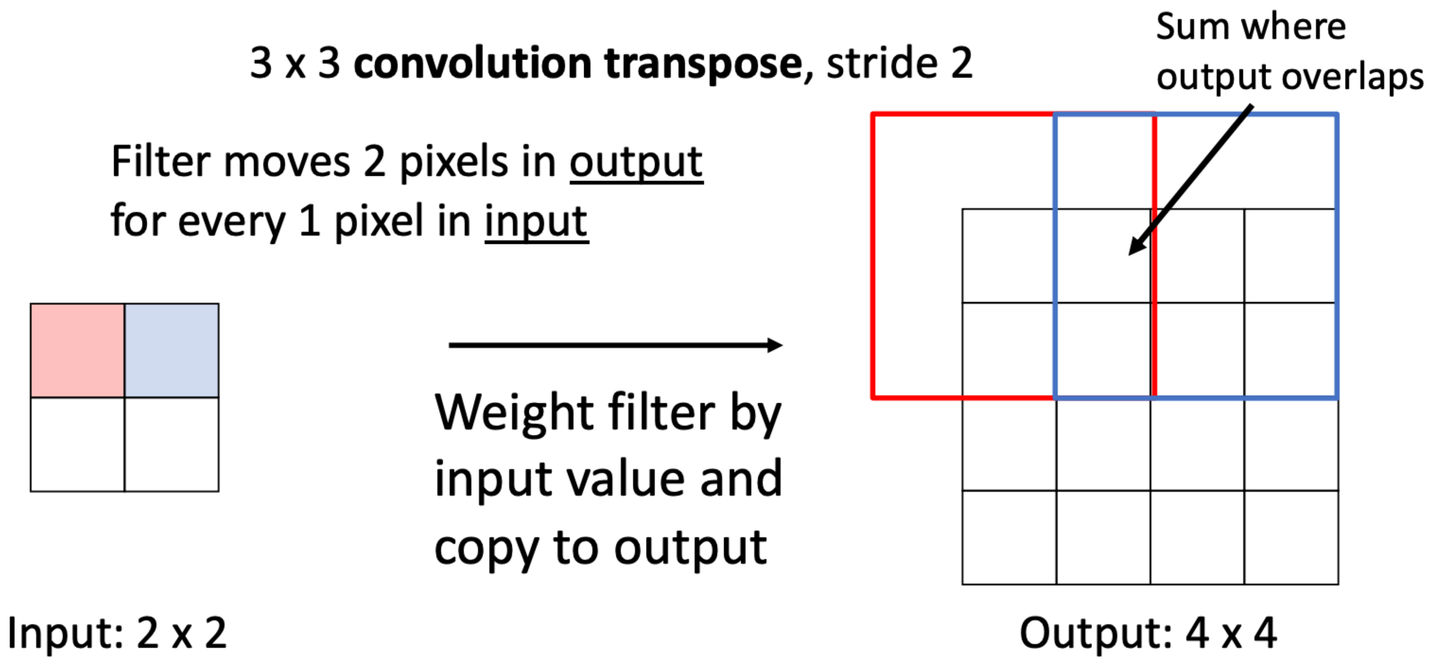
Input: 2 x 2

Weight filter by
input value and
copy to output



Output: 4 x 4

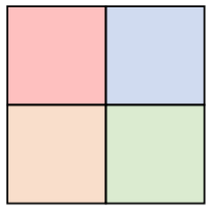
Learnable upsampling: deconvolution



Learnable upsampling: deconvolution

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

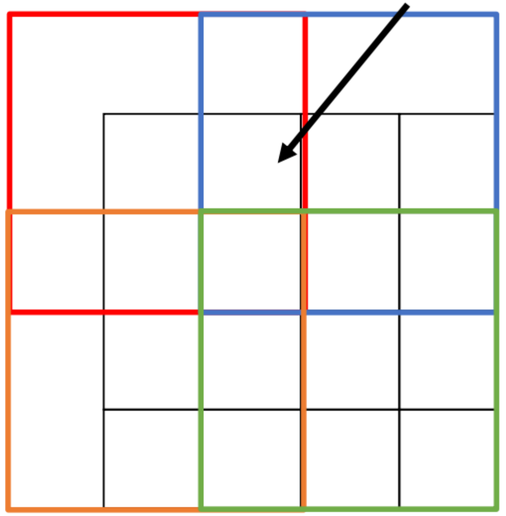
This gives 5x5 output – need to trim one pixel from top and left to give 4x4 output



Input: 2 x 2

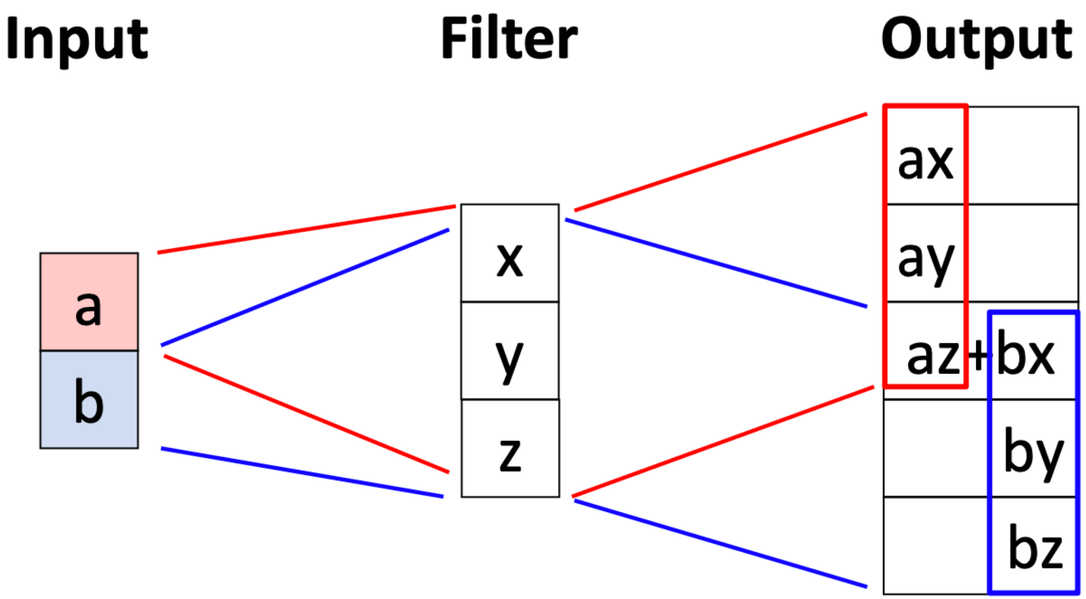
Weight filter by input value and copy to output

Sum where output overlaps



Output: 4 x 4

Learnable upsampling: deconvolution

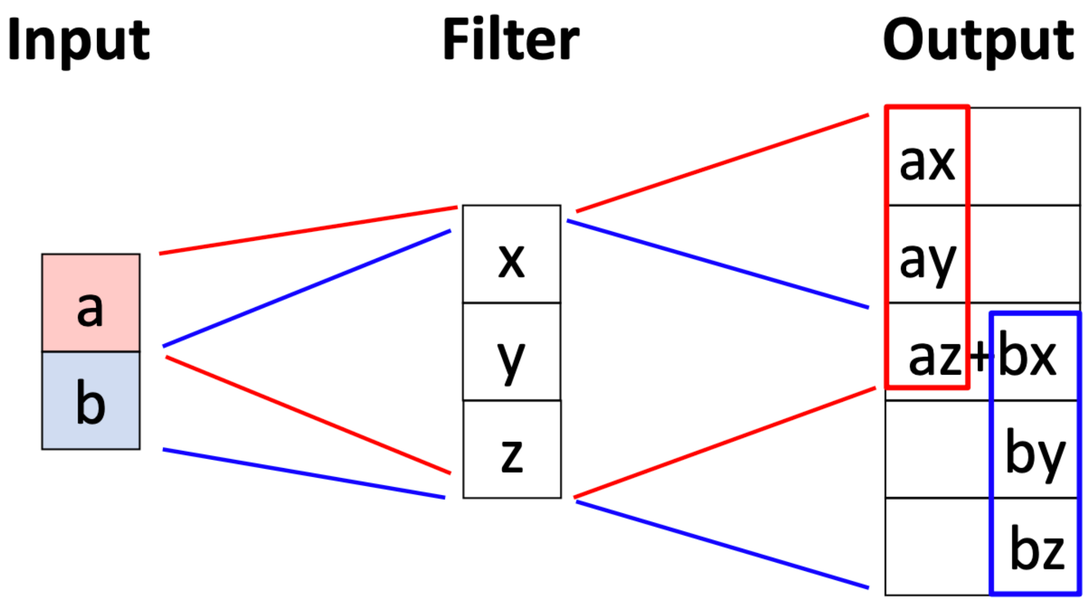


Output has copies of filter weighted by input

Stride 2: Move 2 pixels output for each pixel in input

Sum at overlaps

Learnable upsampling: deconvolution



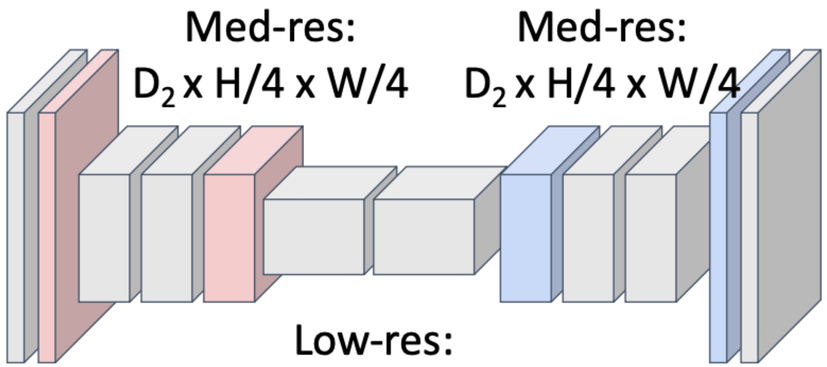
This operation has many names:

- deconvolution
- upconvolution
- fractionally strided convolution
- backward strided convolution
- **transposed convolution**

Semantic segmentation: FCN

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

Input:
3 x H x W

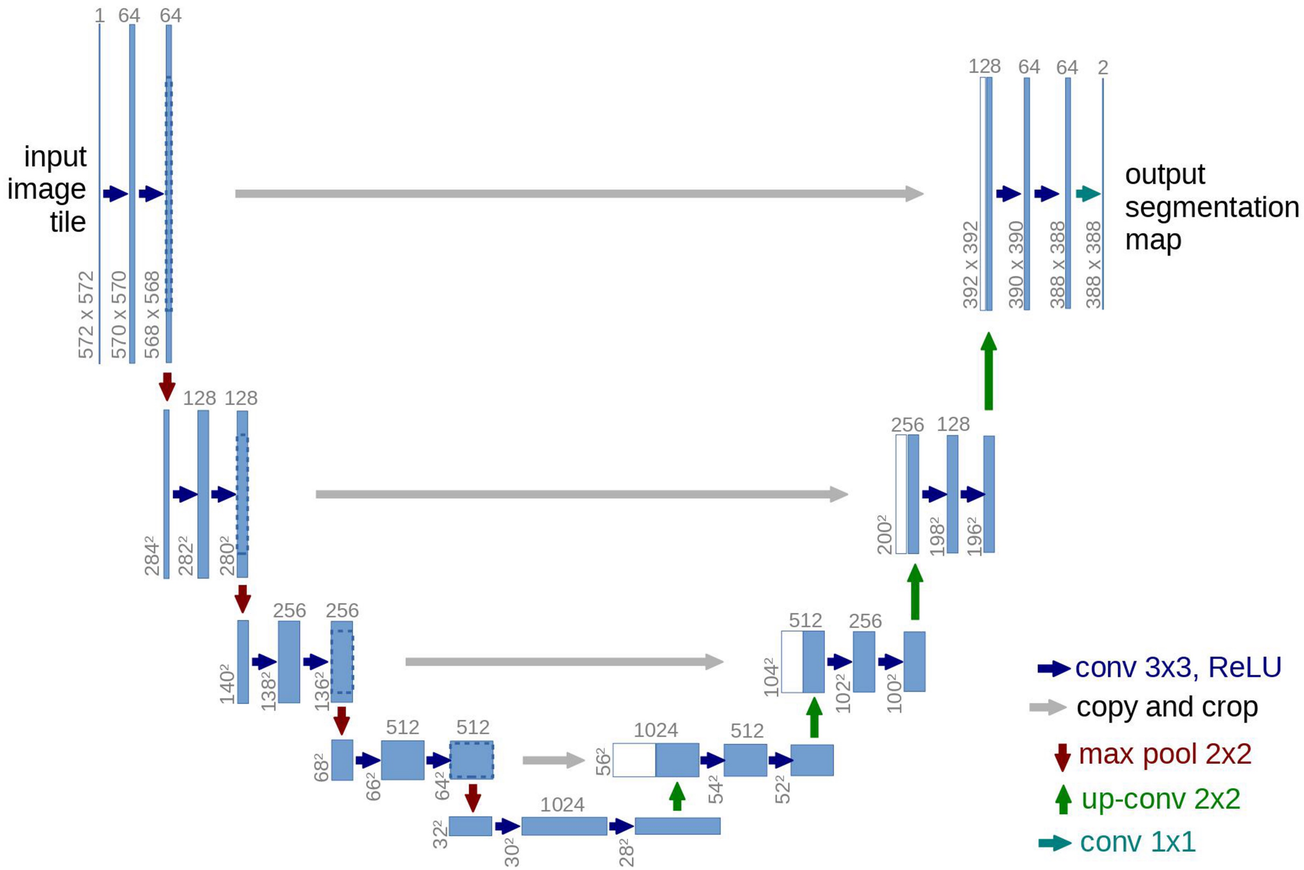


Predictions:
H x W

Downsampling:
pooling, strided convolution

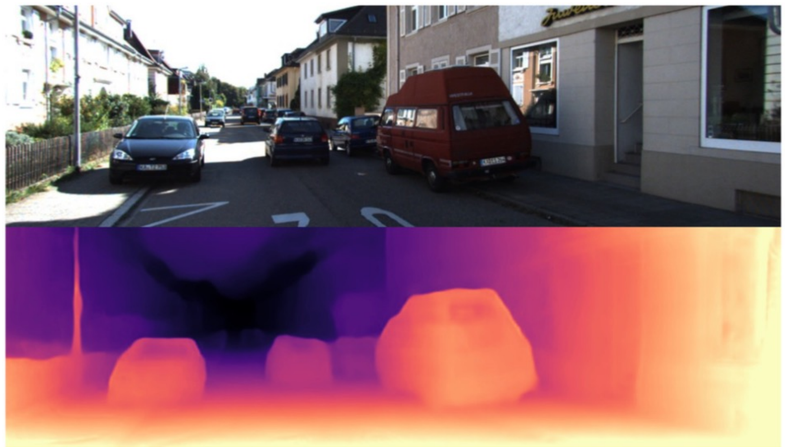
Upsampling:
interpolation, deconvolution (transposed convolution)

Back to UNet

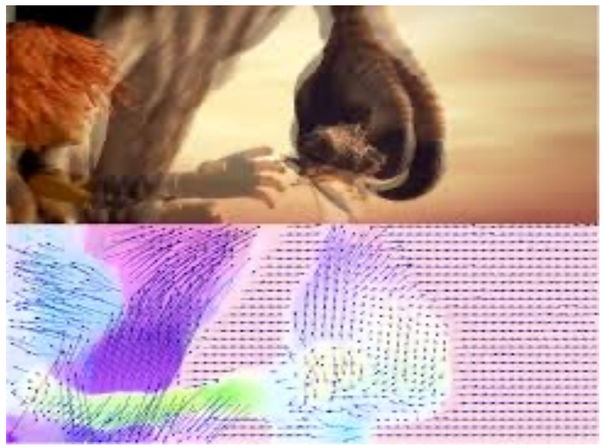

input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Other pixel-level tasks

semantic segmentation



depth estimation

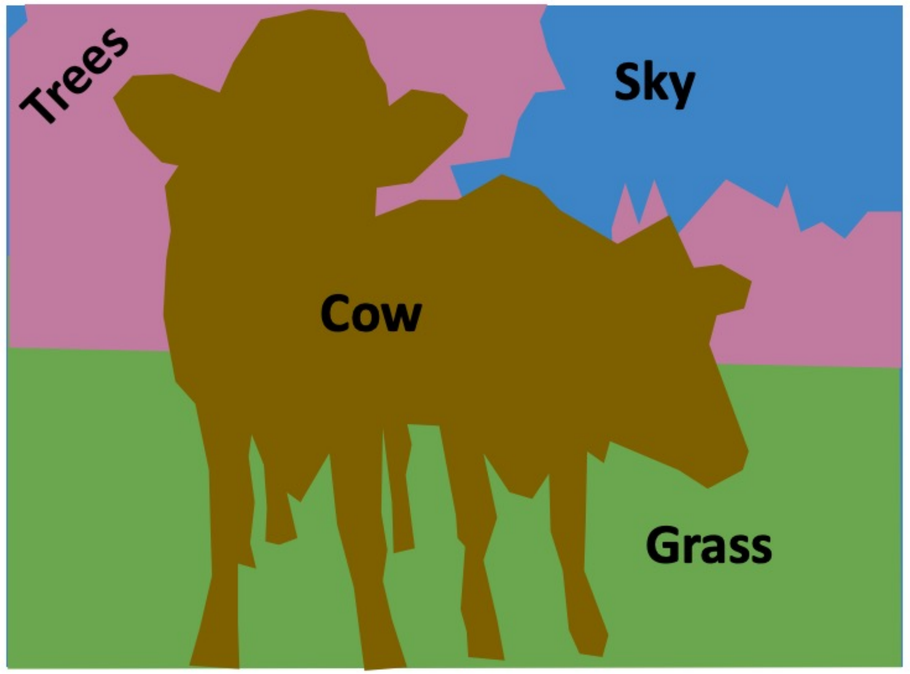


optical flow estimation

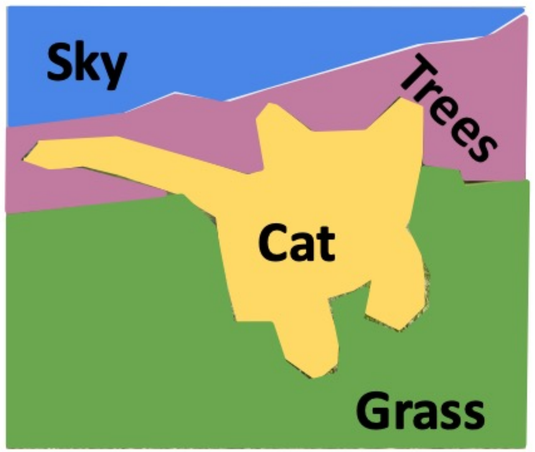
Detection vs semantic segmentation

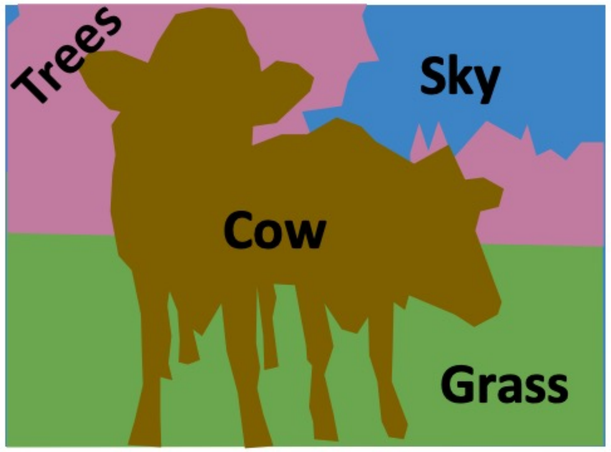
Object detection: Detects individual object instances, but only gives box

Semantic segmentation: Gives per-pixel labels, but merges instances



Things vs Stuff





Things: Object categories that can be separated into object instances (e.g. cat, car, person)

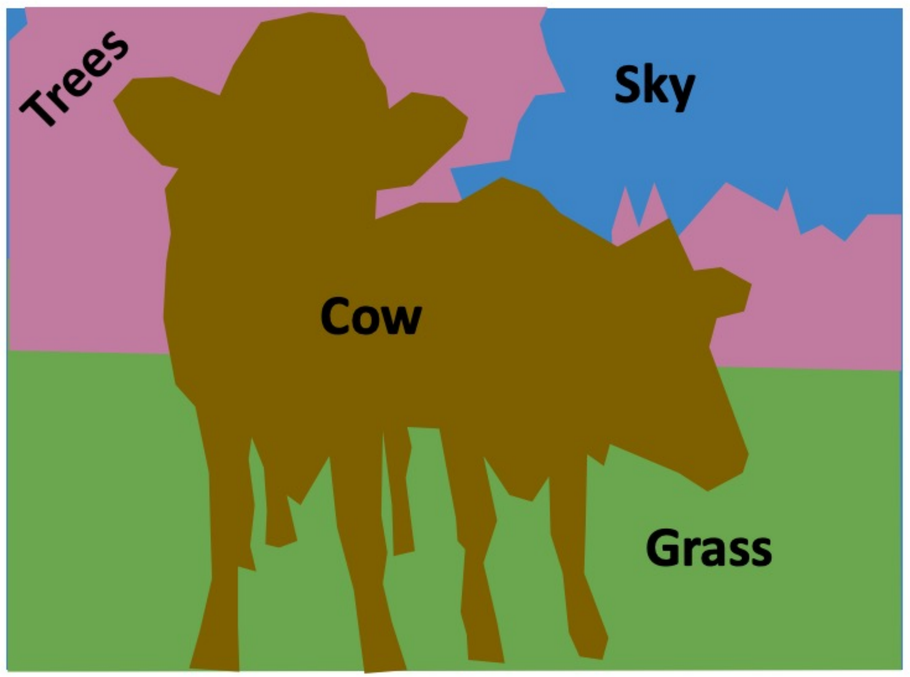
Stuff: Object categories that cannot be separated into instances (e.g. sky, grass, water)

Detection vs semantic segmentation

Object detection: Detects individual object instances, but only gives box

only things!

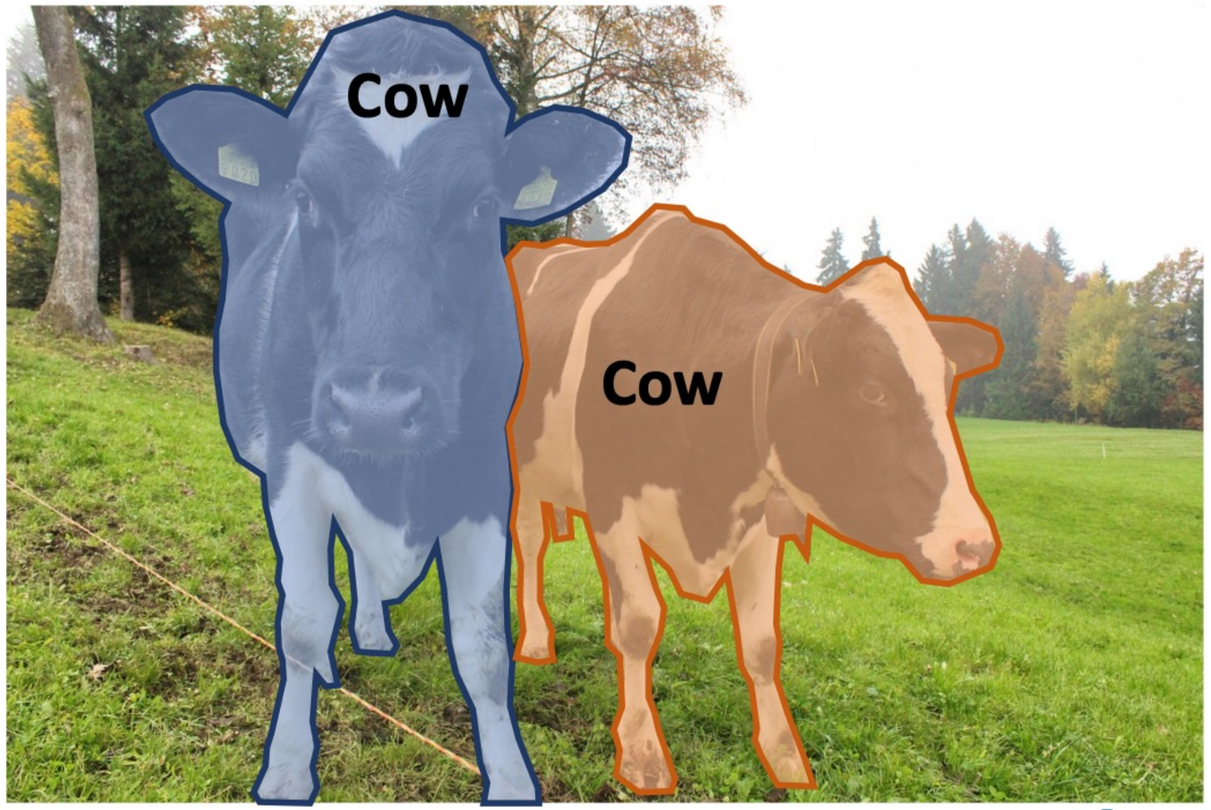
Semantic segmentation: Gives per-pixel labels, but merges instances



both things and stuff

Instance segmentation

Task: Detect all objects in the image, and identify the pixels that belong to each object instance.



only things!

Object detection: Faster-RCNN

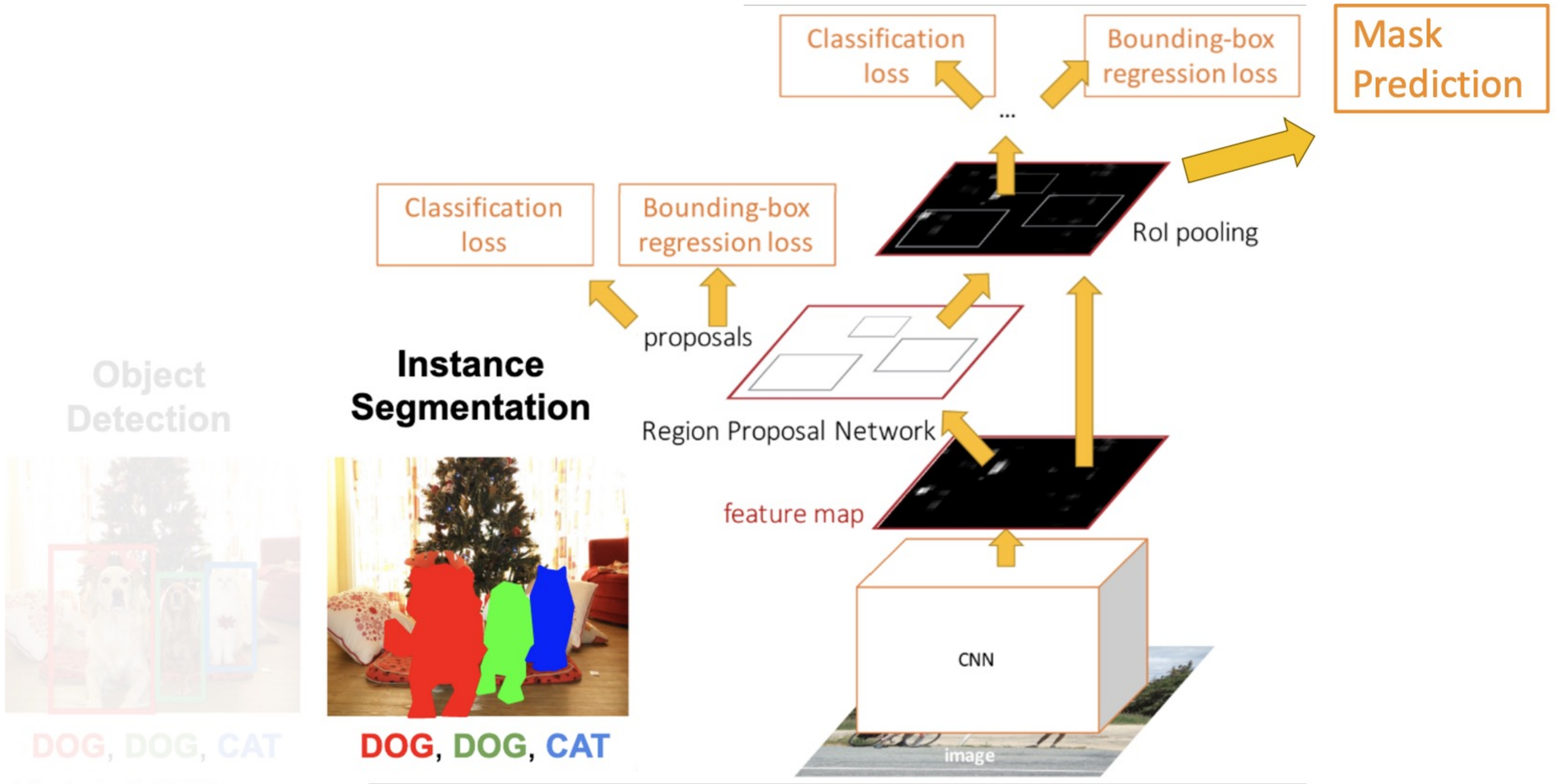
Object Detection

DOG, DOG, CAT

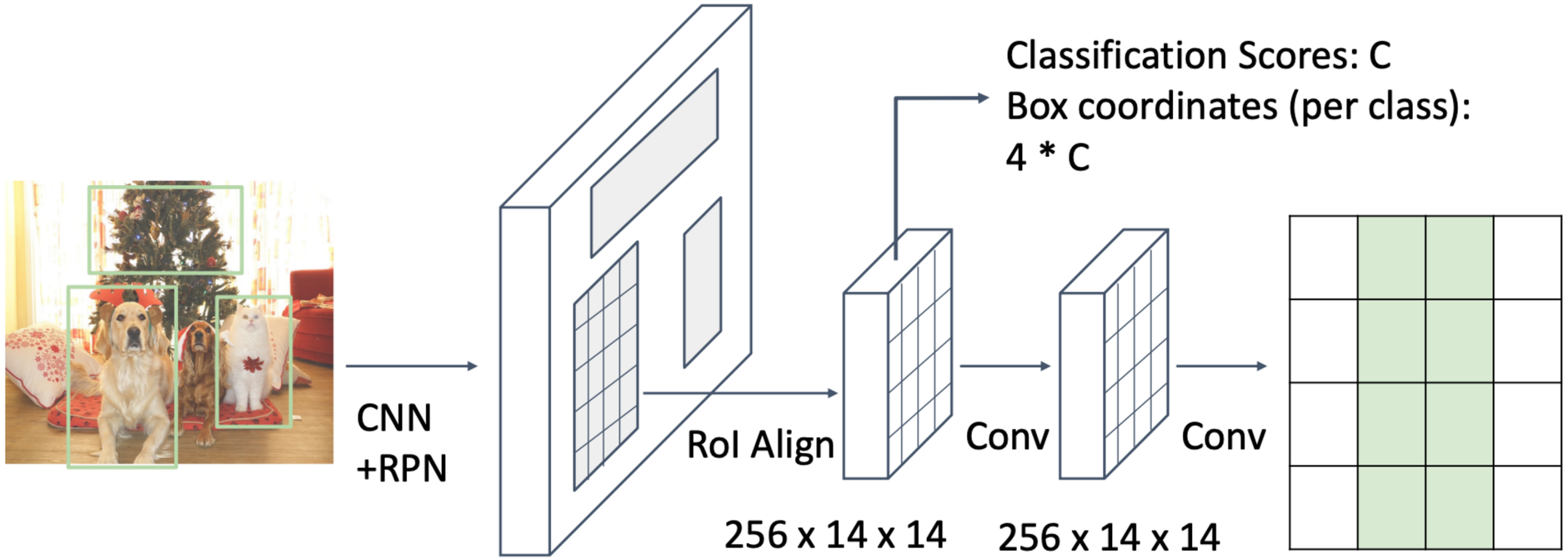
Instance Segmentation

DOG, DOG, CAT

Instance segmentation: Mask-RCNN

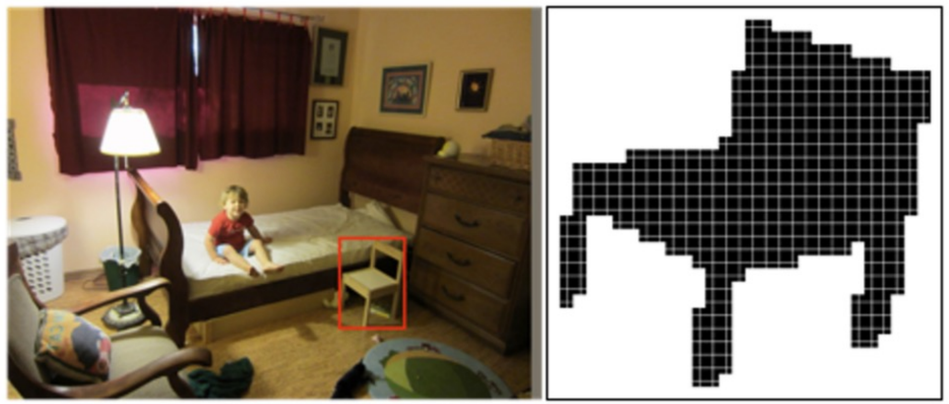


Mask-RCNN

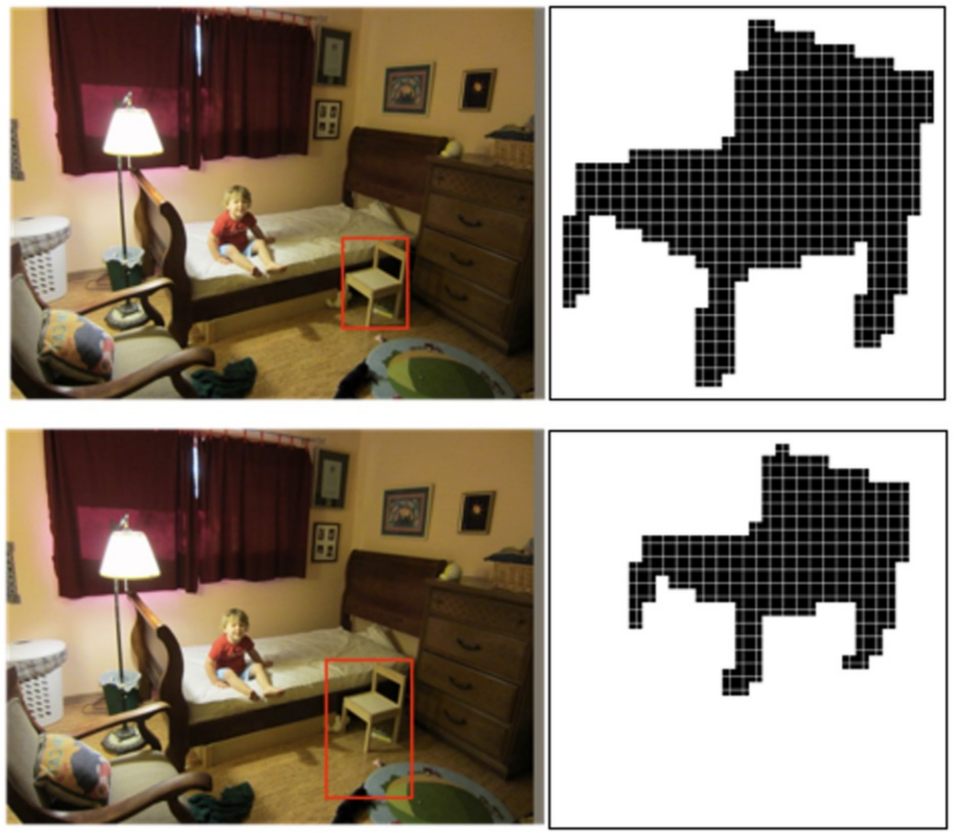


Idea: Predict a mask for each of C classes: C x 28 x 28

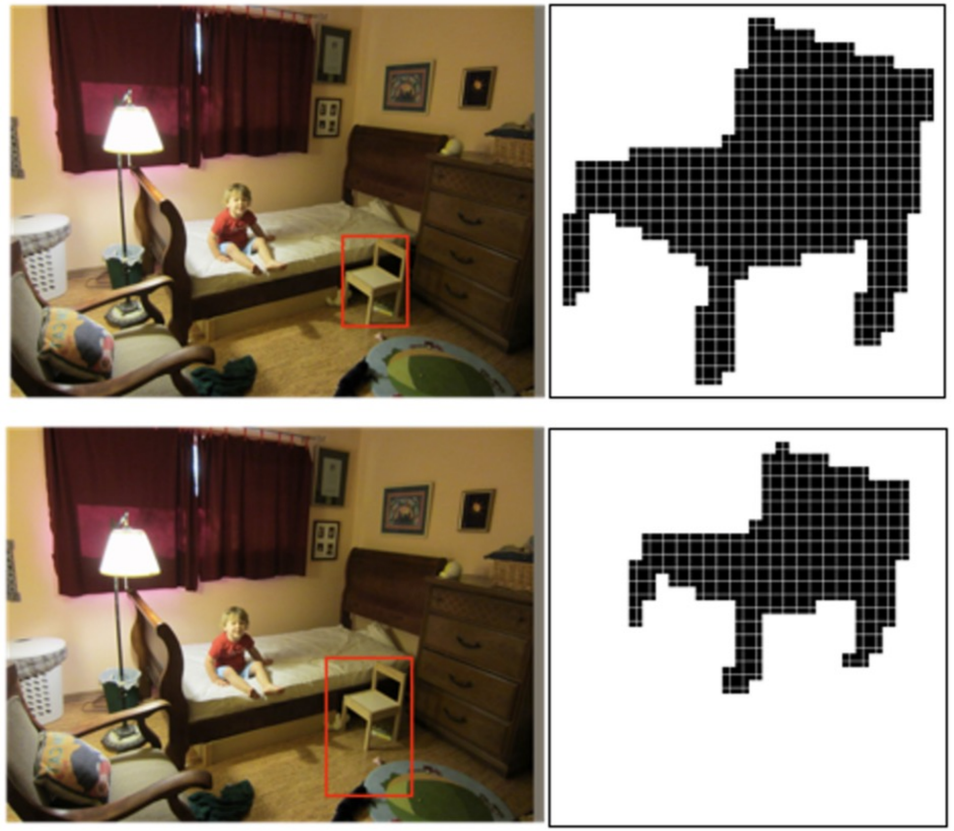
Mask-RCNN: Example Training Targets



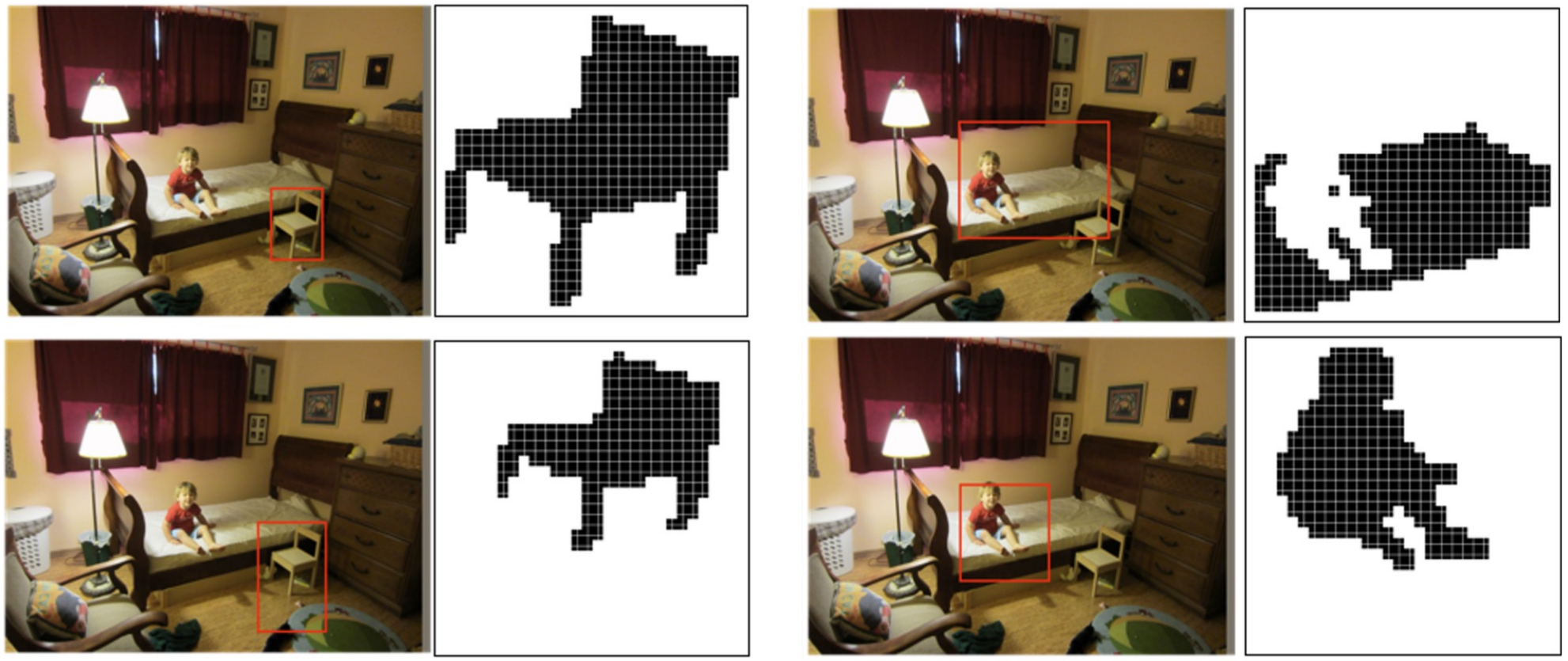
Mask-RCNN: Example Training Targets



Mask-RCNN: Example Training Targets

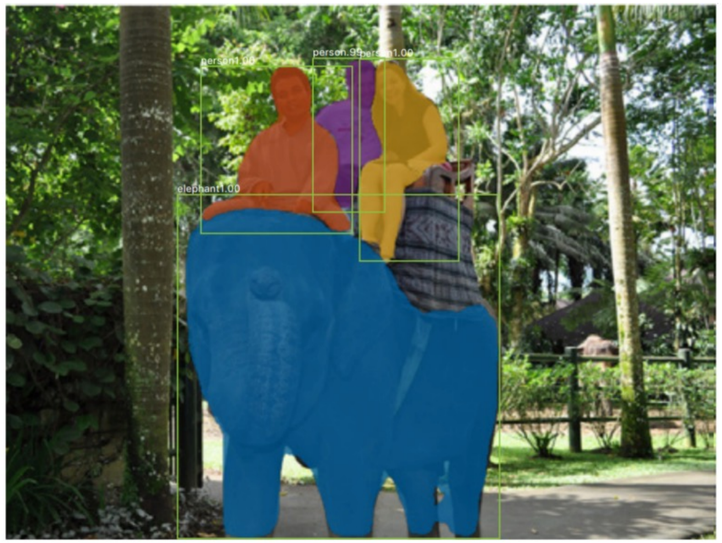


Mask-RCNN: Example Training Targets



Mask-RCNN

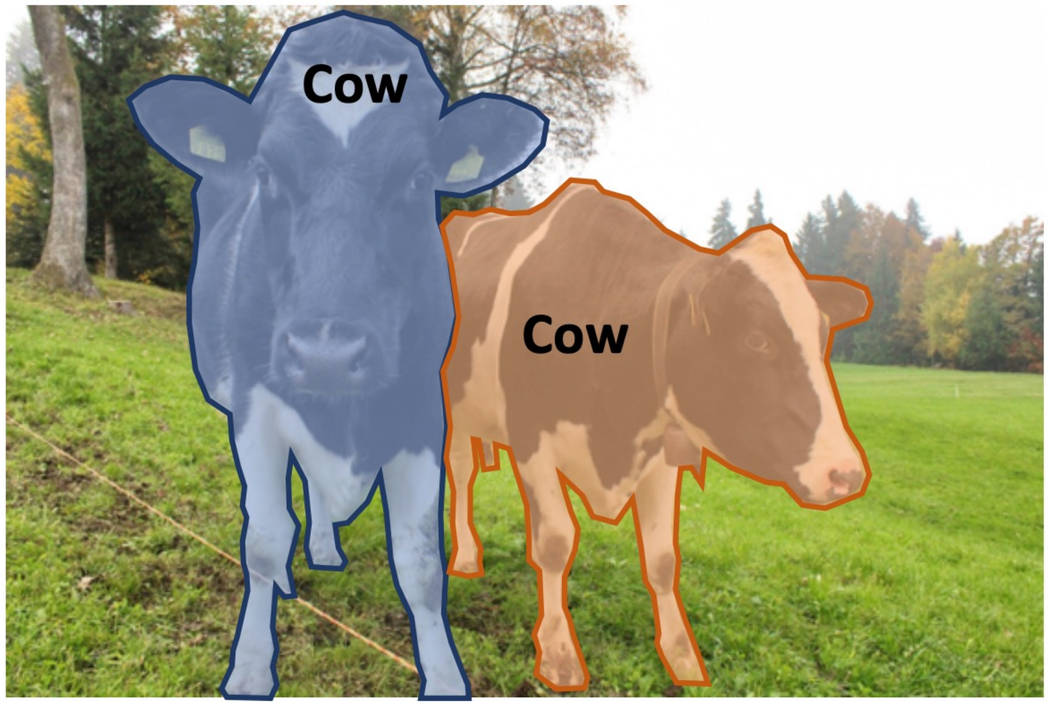

person1.00
person1.00
motorcycle1.00
motorcycle1.00
person.96
person1.00
person.92


person1.00
elephant1.00

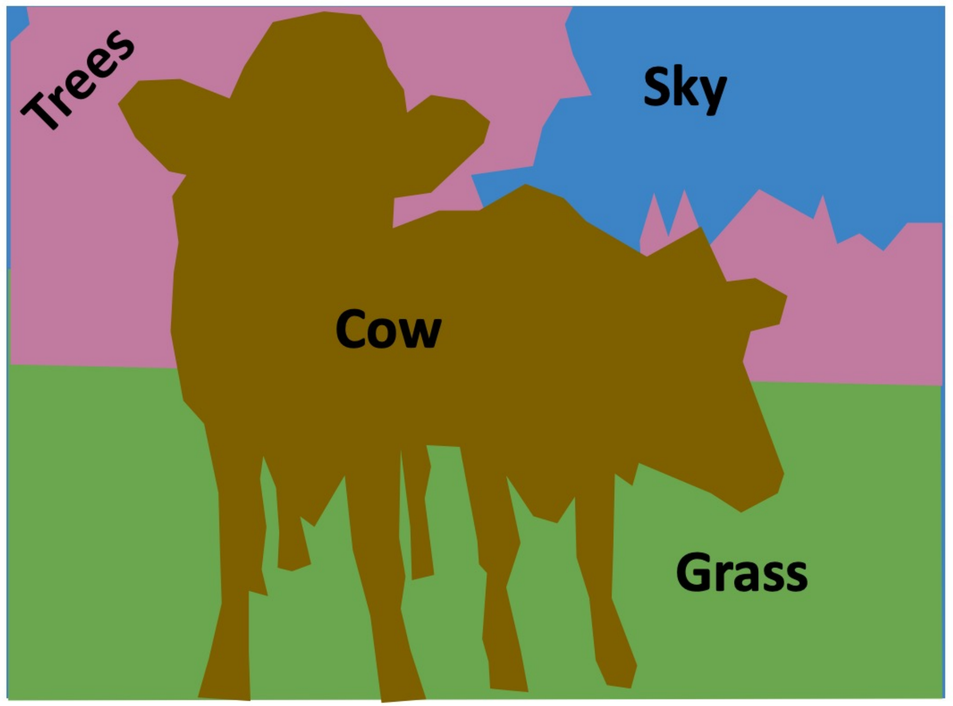

person1.00
person1.00
person1.00
person1.00
dining table.96
bottle.99
bottle.99
bottle.99

Beyond instance segmentation

Instance segmentation: Separate object instances, but only things



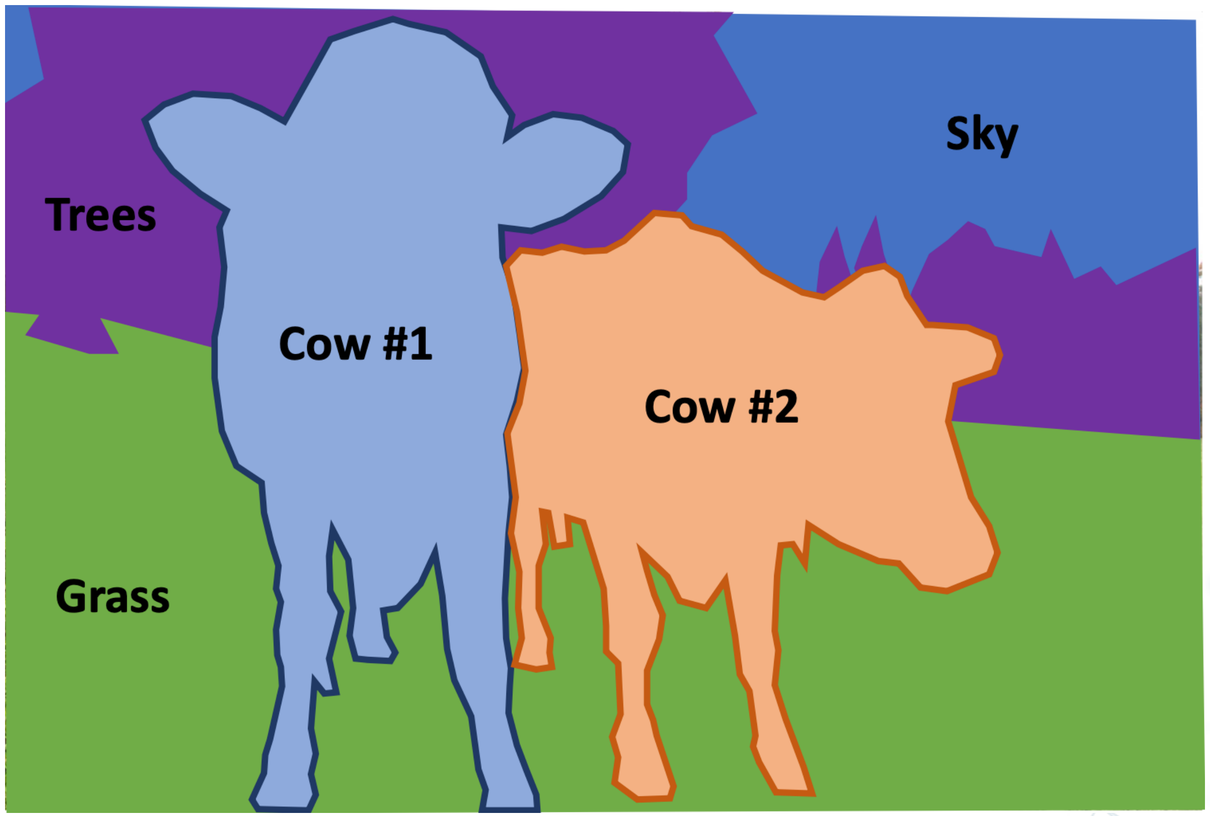
Semantic segmentation: Identify both things and stuff, but doesn't separate instances



Panoptic segmentation

Panoptic segmentation task:
Label all pixels in the image (both things and stuff)

For “things” categories also separate into instances.



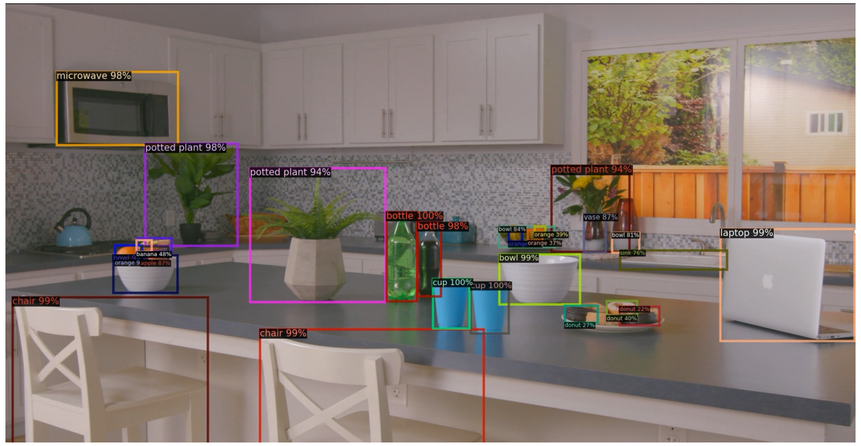
Kirillov et al, “Panoptic Segmentation”, CVPR 2019
Kirillov et al, “Panoptic Feature Pyramid Networks”, CVPR 2019

Panoptic segmentation

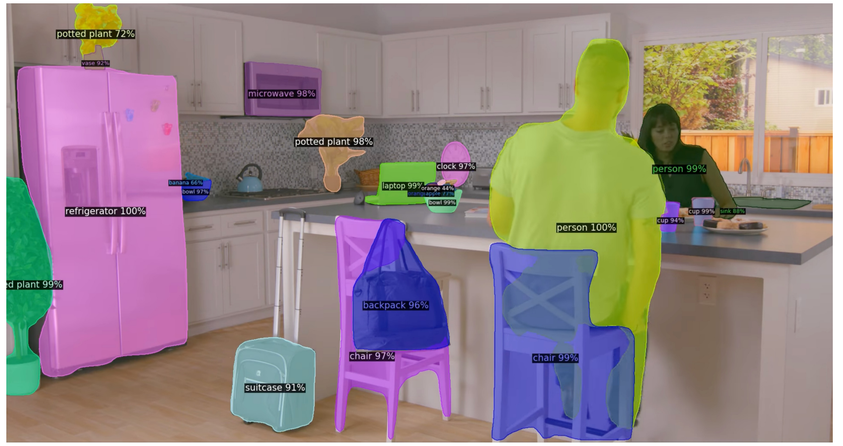
Kirillov et al, “Panoptic Segmentation”, CVPR 2019
Kirillov et al, “Panoptic Feature Pyramid Networks”, CVPR 2019

Visual perception – instance-level

bounding box detection



instance segmentation



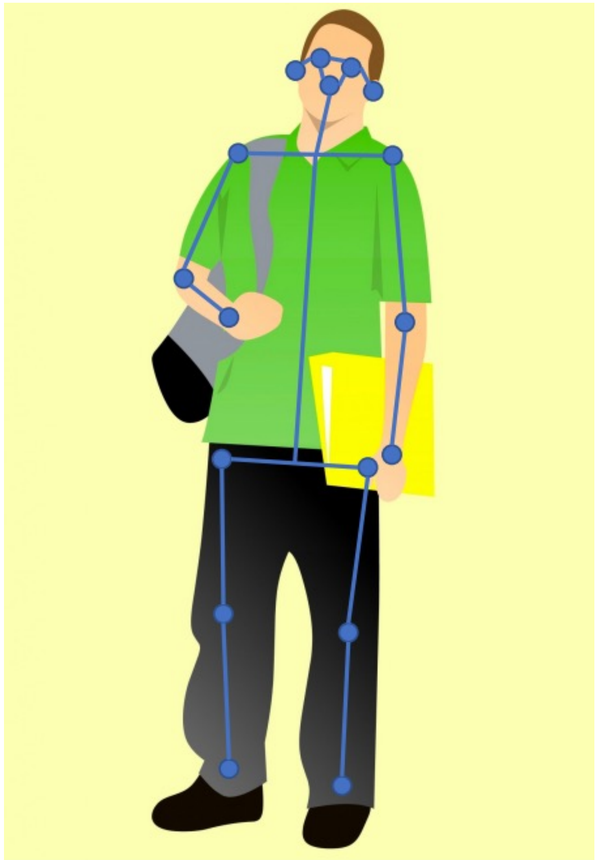
pose estimation

dense correspondences

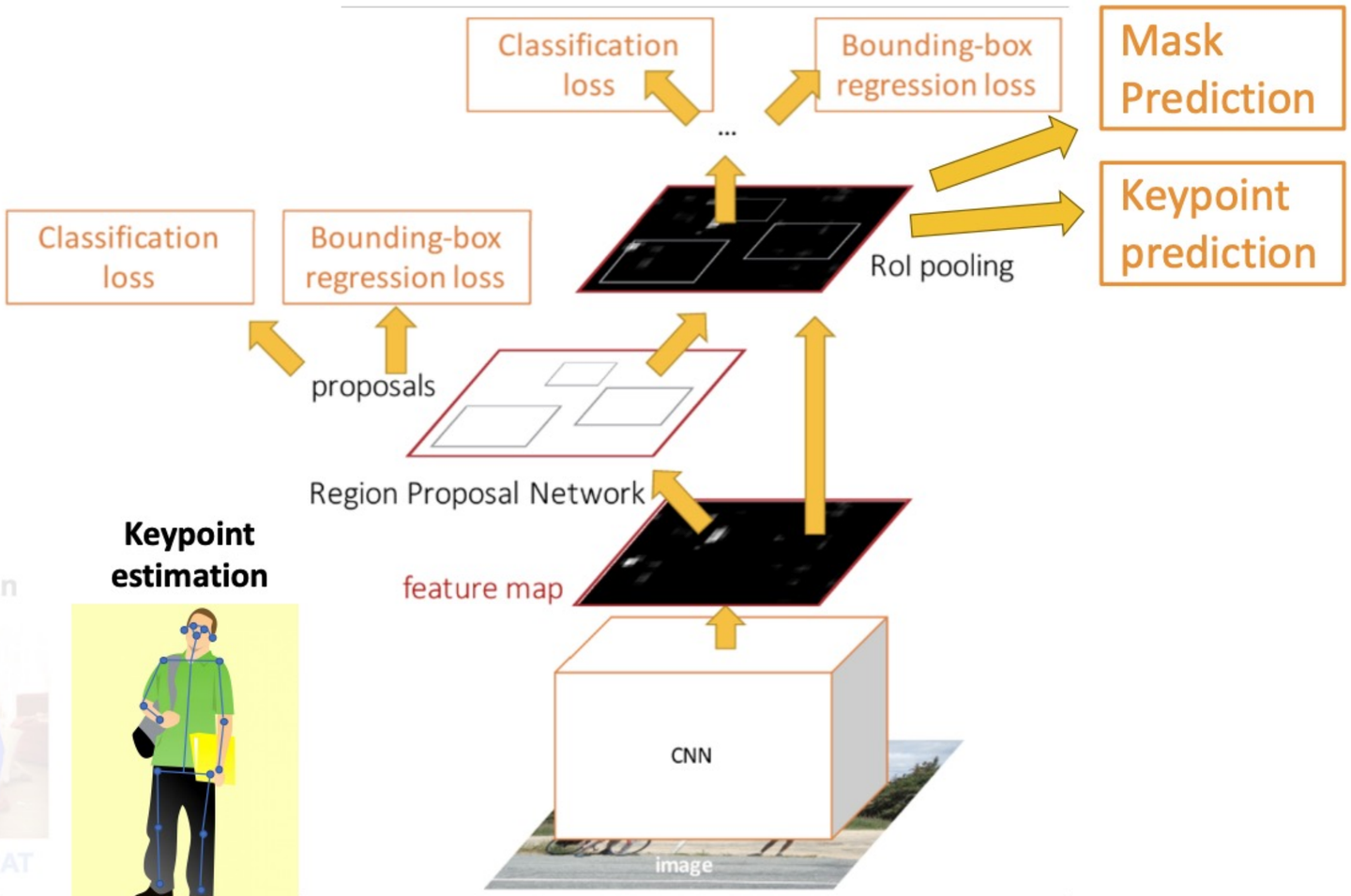
Human pose estimation

Represent the pose of a human by locating a set of 17 keypoints:

- Nose
- Left / Right eye
- Left / Right ear
- Left / Right shoulder
- Left / Right elbow
- Left / Right wrist
- Left / Right hip
- Left / Right knee
- Left / Right ankle



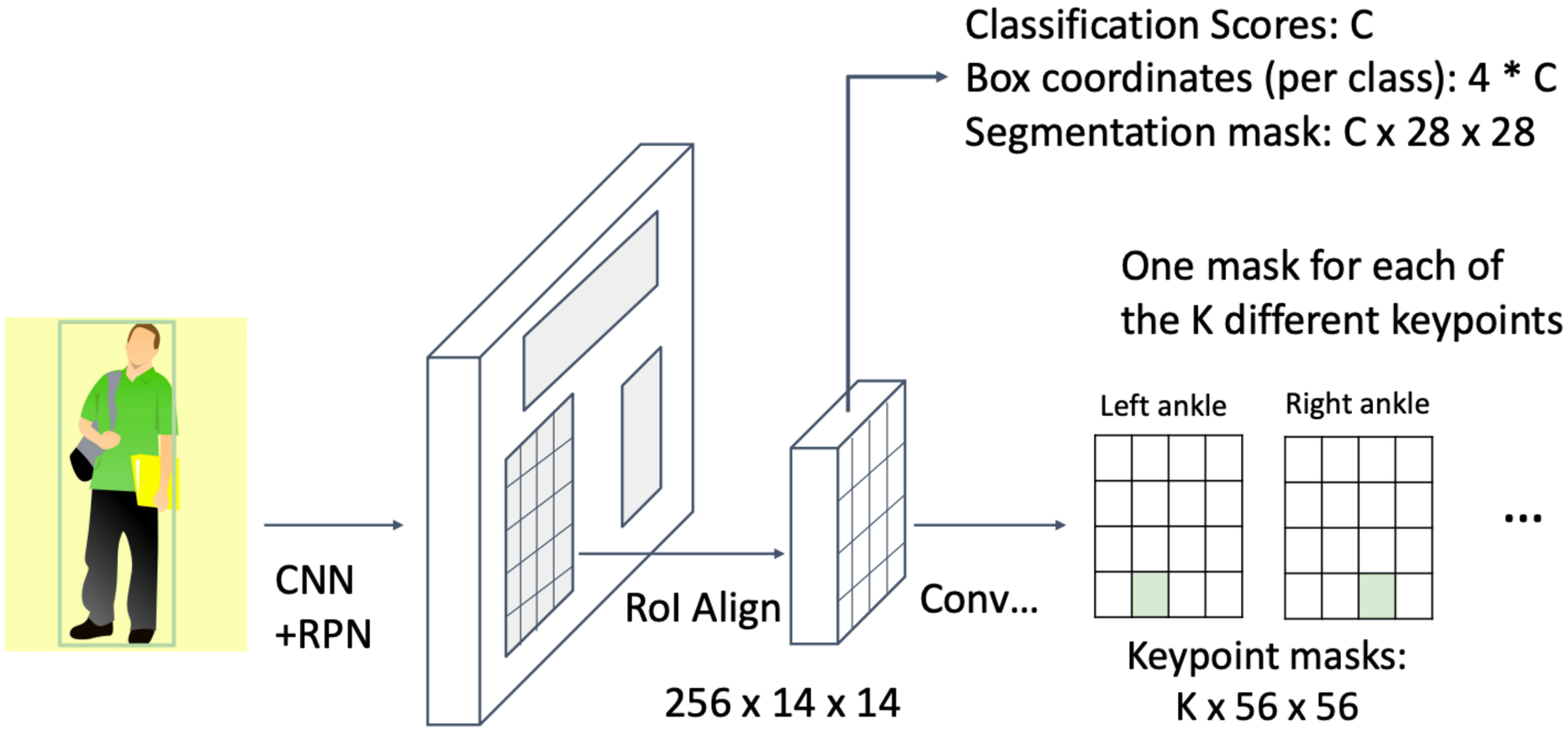
Mask-RCNN: keypoint estimation



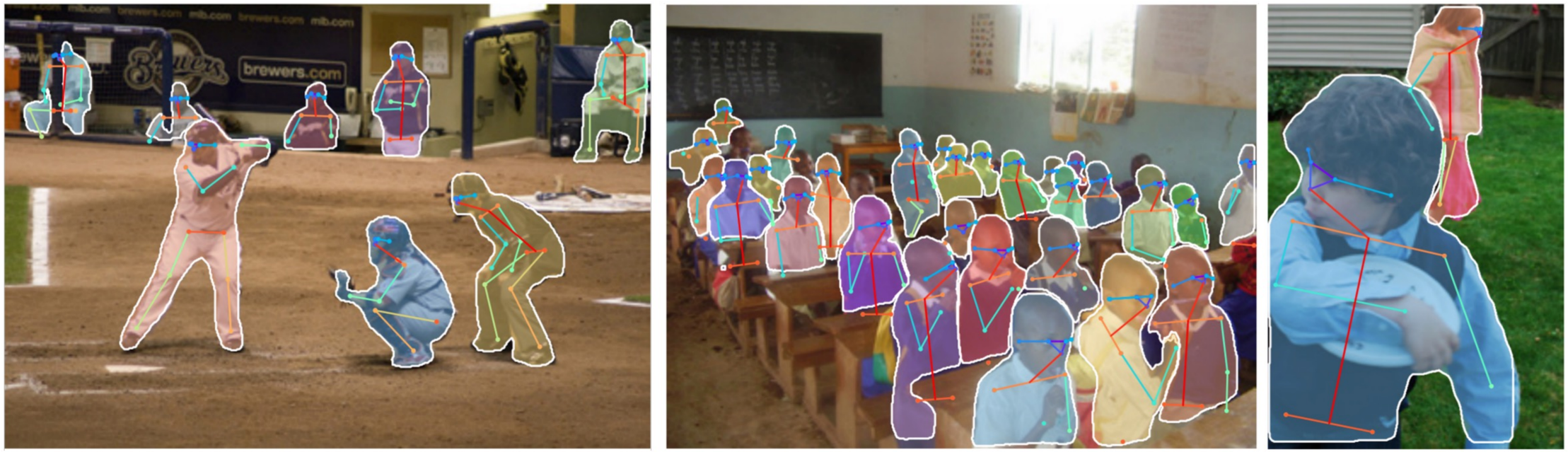
Keypoint estimation

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

Mask-RCNN: keypoint estimation

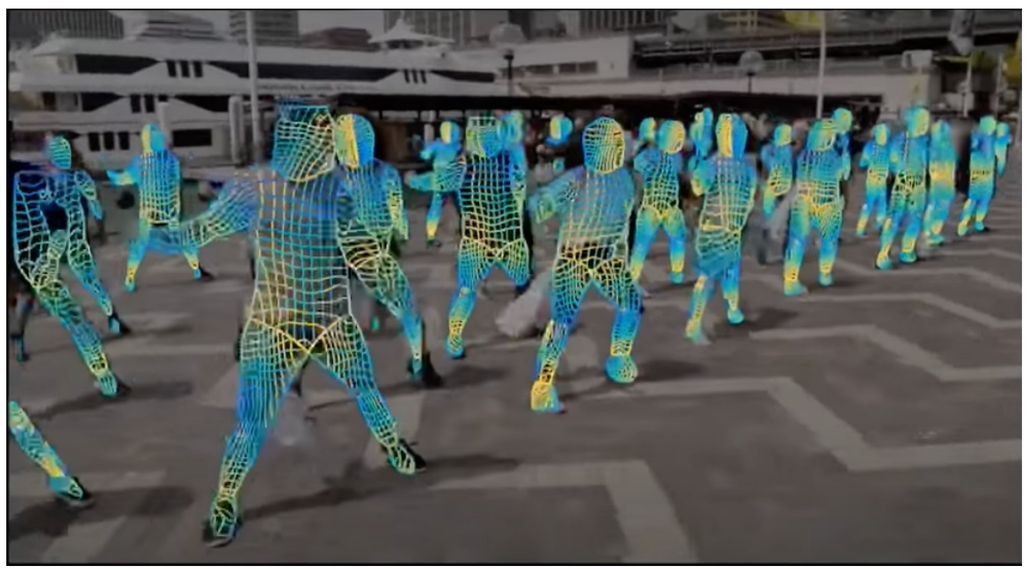


Joint Instance Segmentation & Keypoint Estimation



Add more heads!

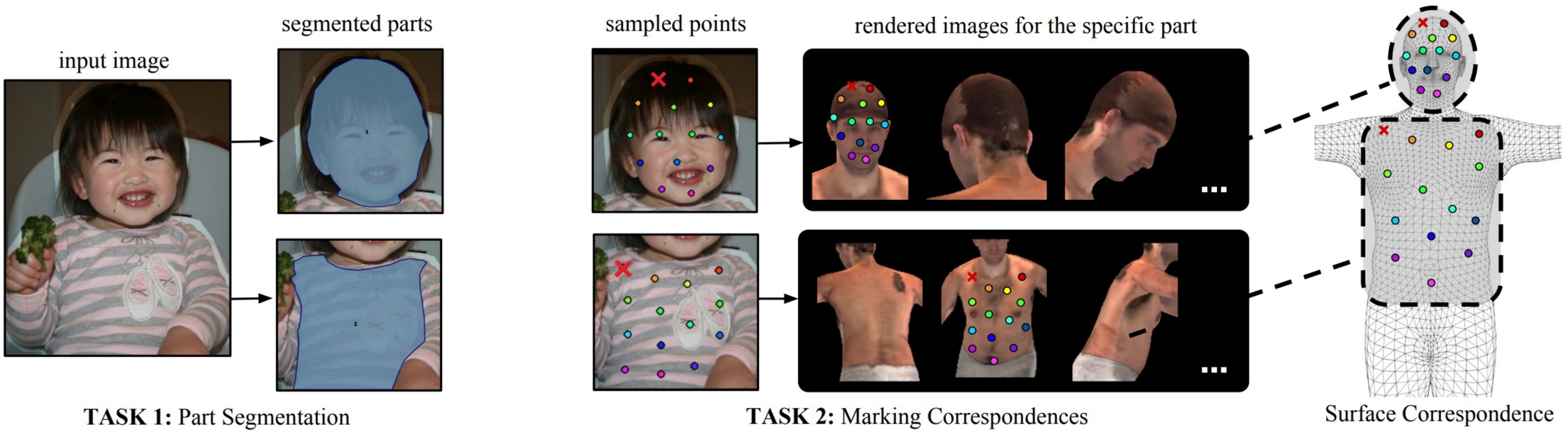
Dense correspondences (DensePose): localize every point on an object surface w.r.t. reference model



Guler, Neverova, Kokkinos, "DensePose: Dense Human Pose Estimation in the Wild". CVPR, 2018
Neverova, et al. Continuous Surface Embeddings. NeurIPS, 2020

Add more heads!

Dense correspondences (DensePose) annotations

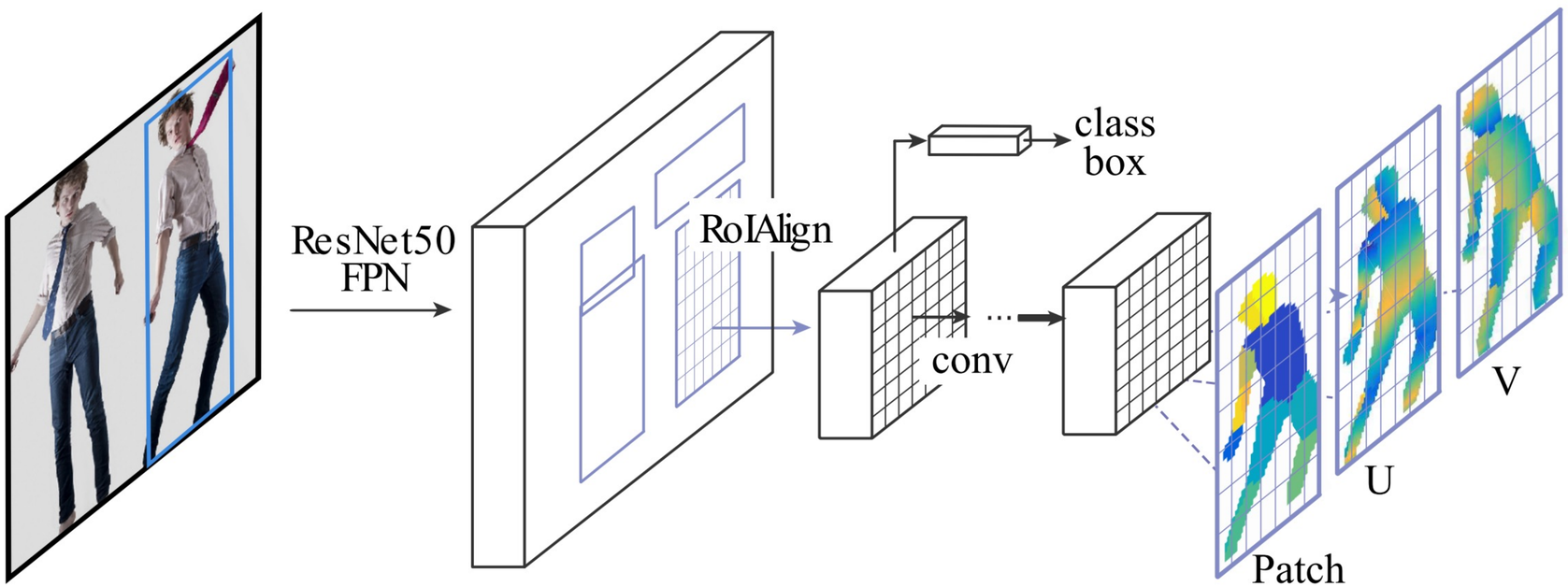


TASK 1: Part Segmentation

TASK 2: Marking Correspondences

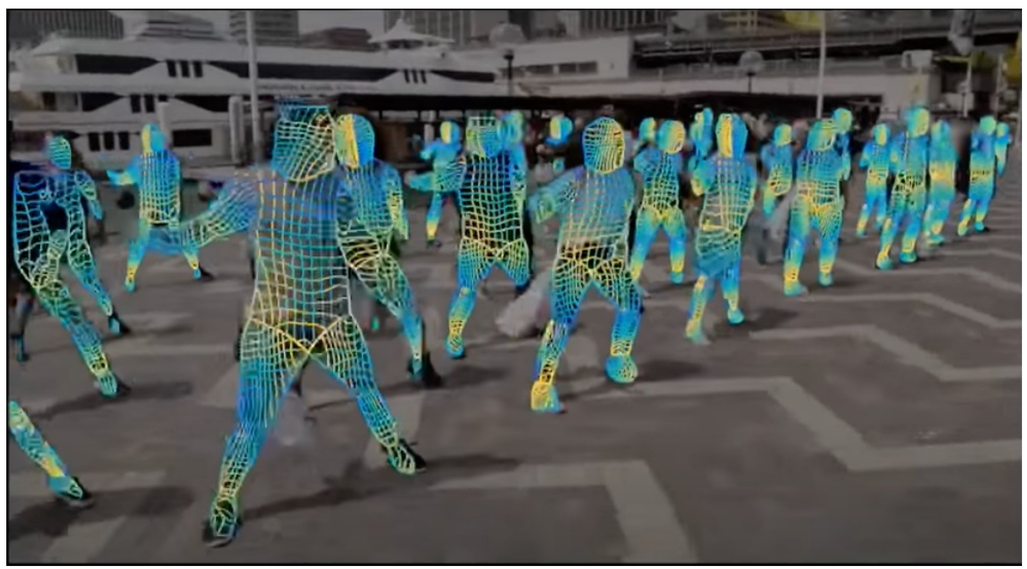
Add more heads!

Dense correspondences (DensePose): localize every point on an object surface w.r.t. reference model



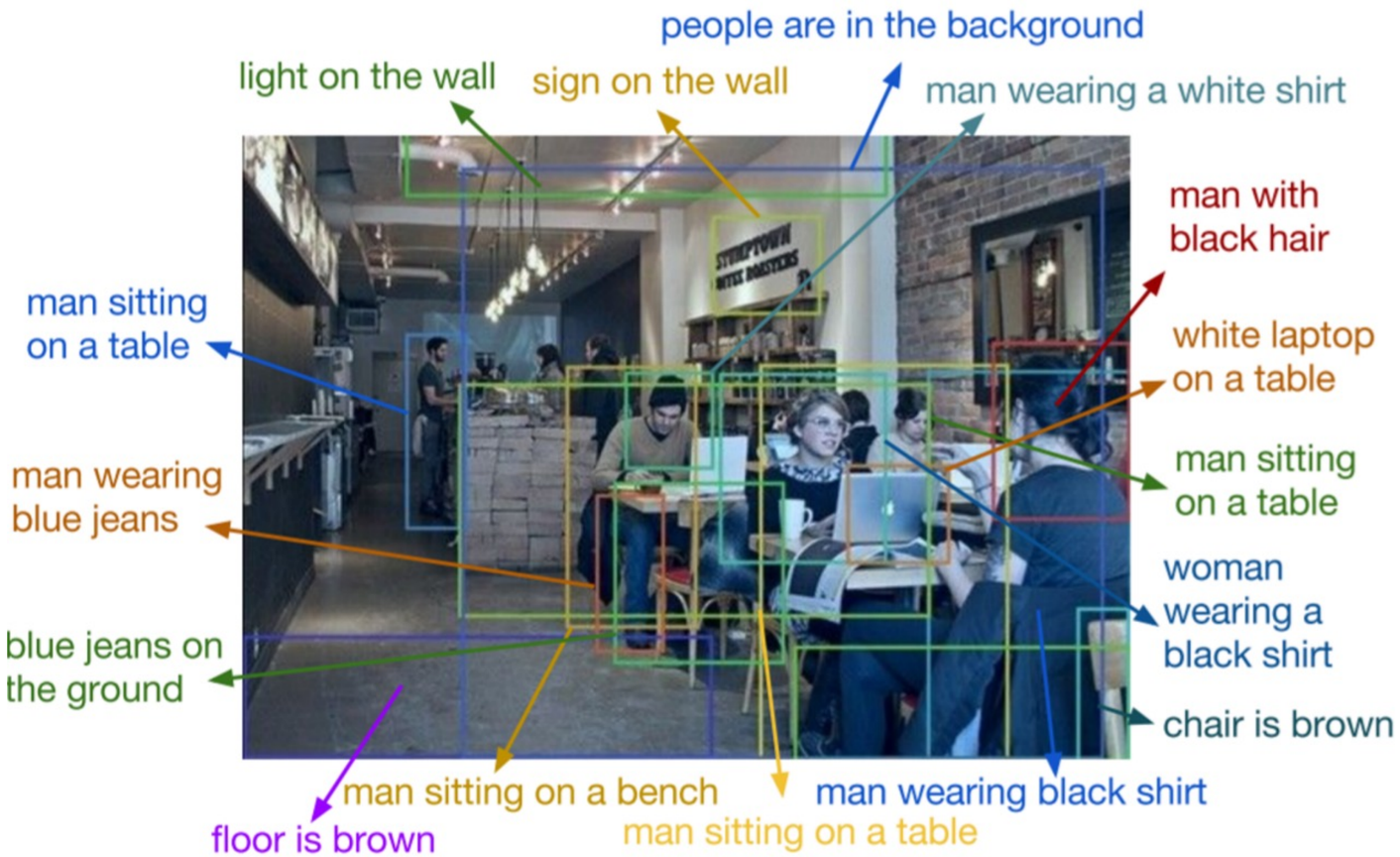
Add more heads!

Dense correspondences (DensePose): localize every point on an object surface w.r.t. reference model



Guler, Neverova, Kokkinos, "DensePose: Dense Human Pose Estimation in the Wild". CVPR, 2018
Neverova, et al. Continuous Surface Embeddings. NeurIPS, 2020

Add more heads!

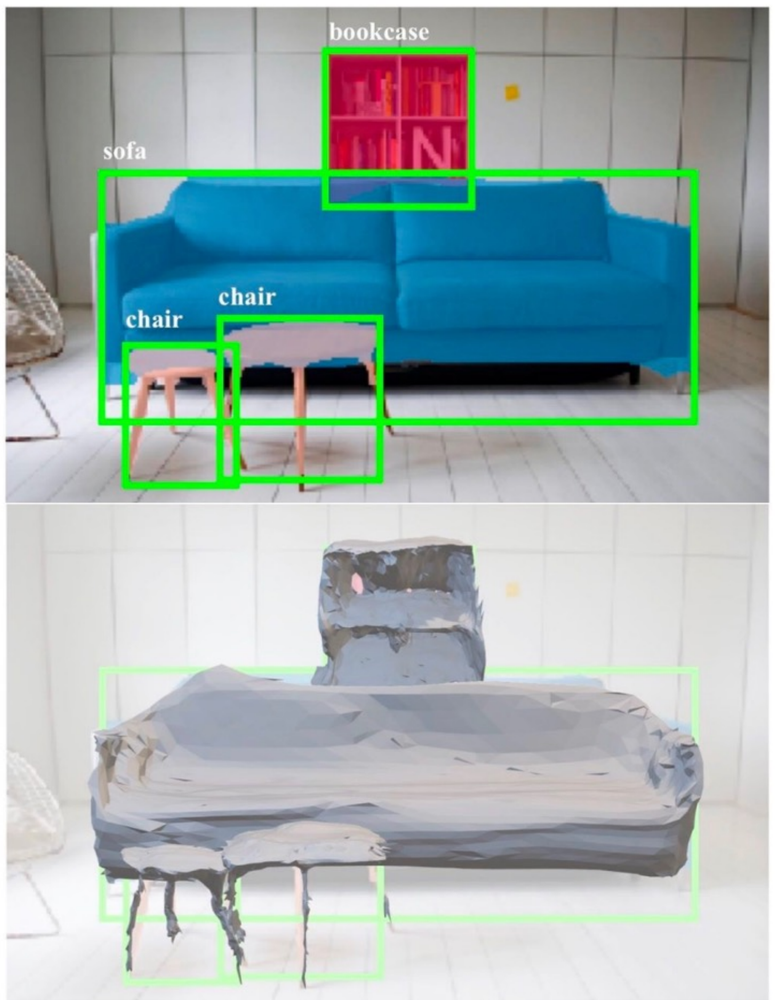


Dense captioning:

predict text description for each object

Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016

Add more heads!

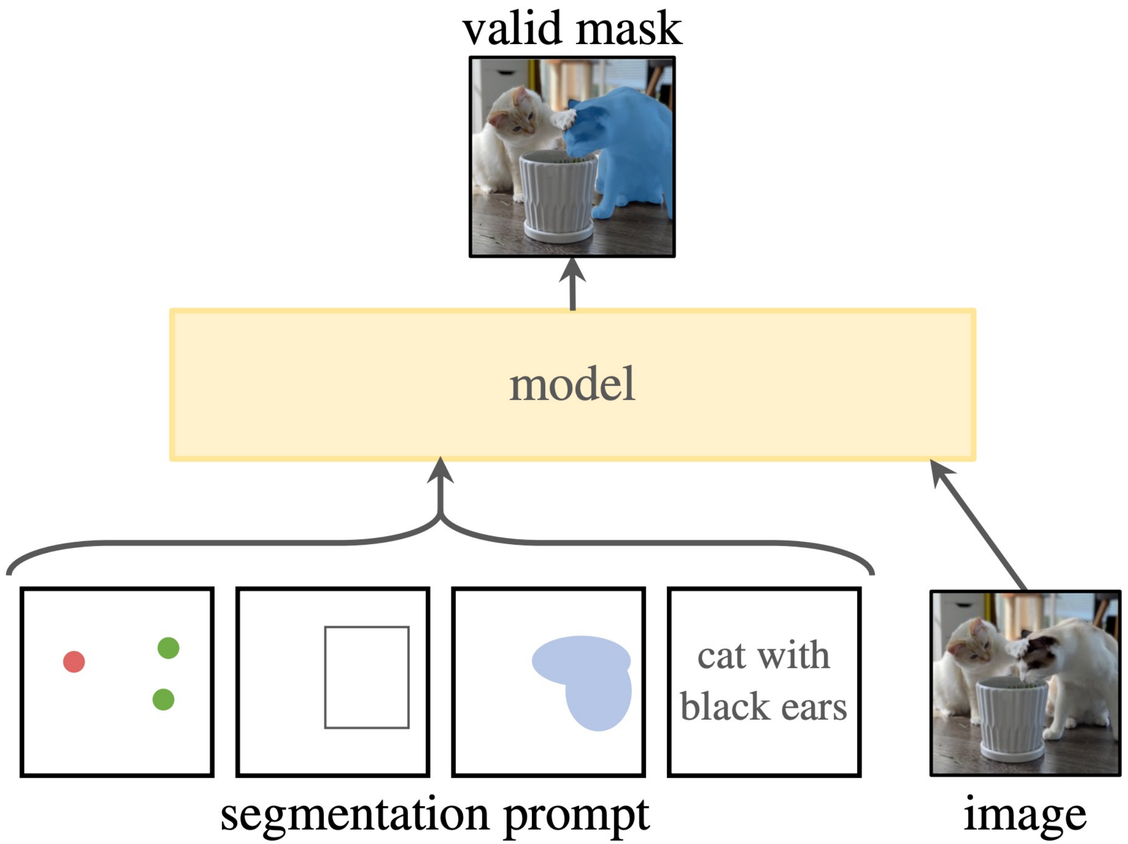


3D reconstruction: predict 3D shape for each object

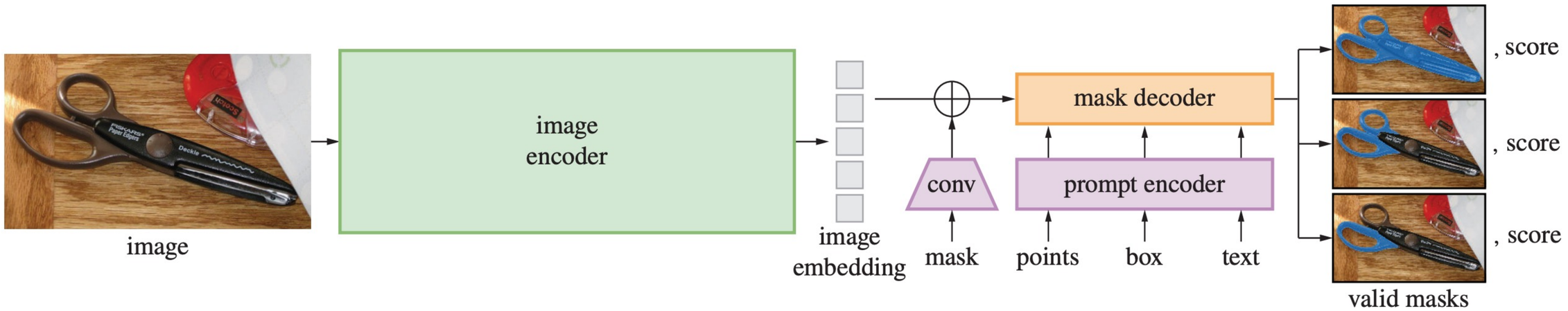
Gkioxari, Malik, and Johnson, "Mesh R-CNN", ICCV 2019

Segment Anything: real-time interactive inference

Segment Anything: promptable segmentation



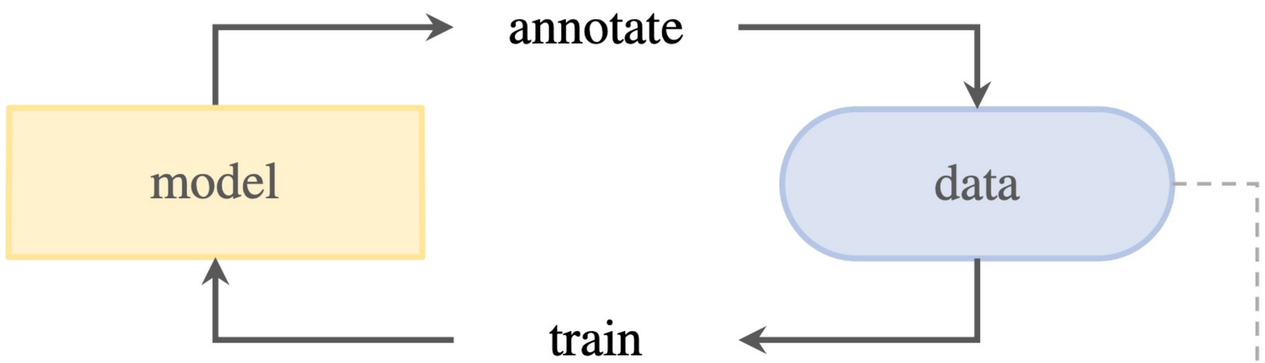
Segment Anything: architecture



The model can be queried by a variety of input prompts to produce masks.

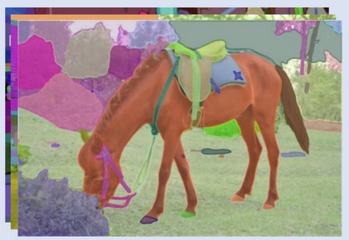
For ambiguous prompts corresponding to more than one object,
compute multiple valid masks with confidence scores.

Segment Anything: dataset



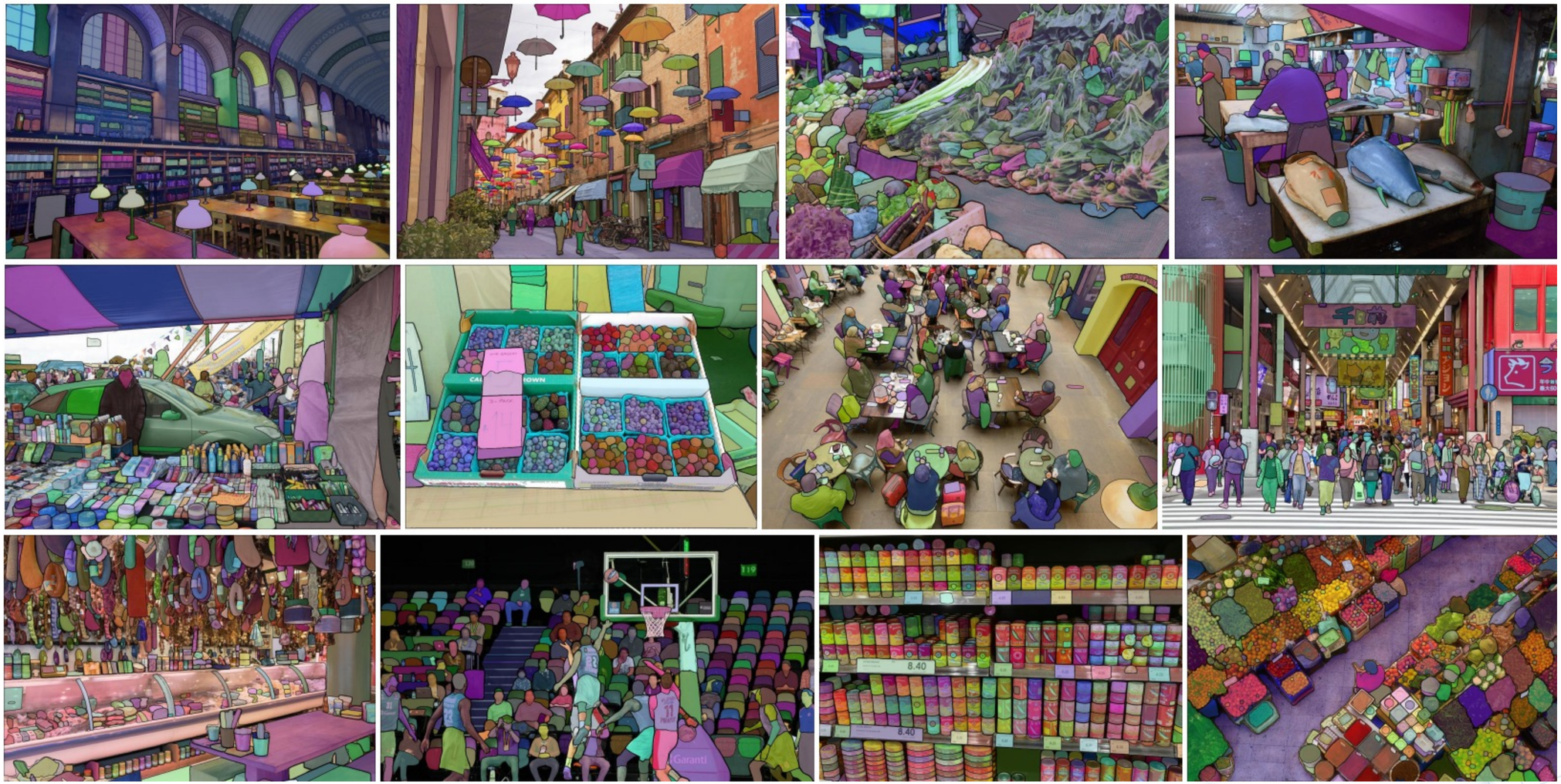
Segment Anything 1B (**SA-1B**):

- **1+ billion masks**
- 11 million images
- privacy respecting
- licensed images



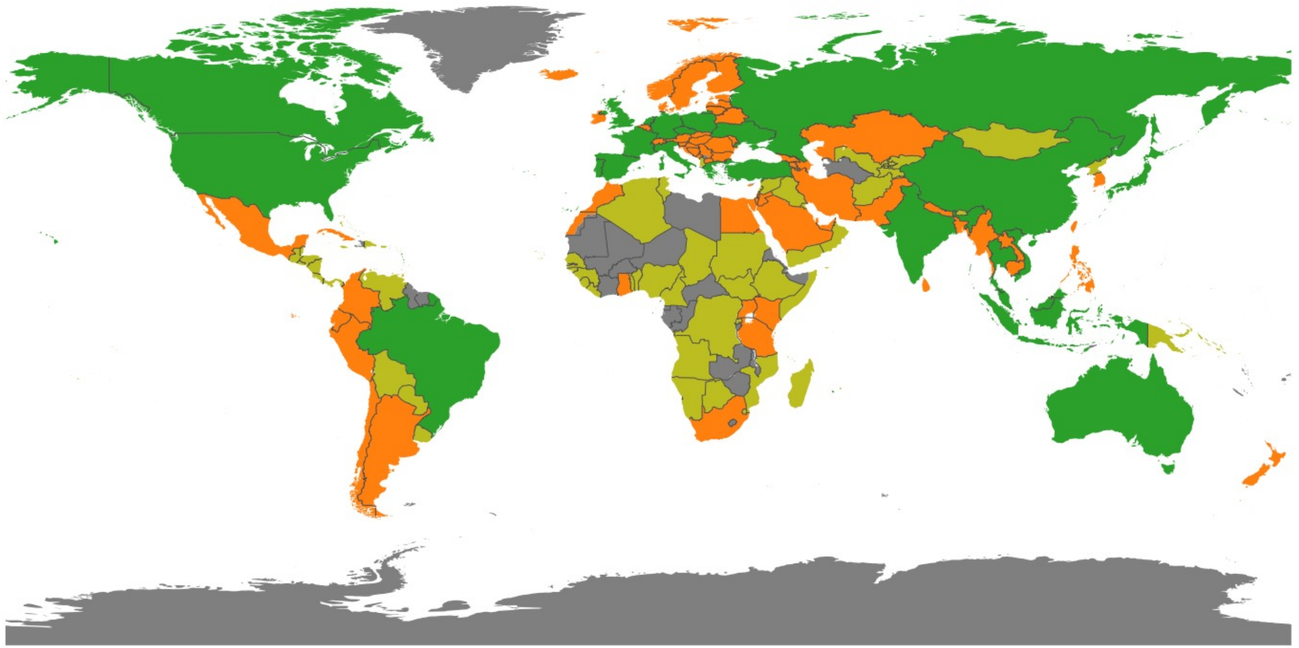
Segment Anything: dataset

Segment Anything: dataset



Segment Anything: dataset

Segment Anything: dataset



Segment Anything: extendable to other tasks

Can take input prompts from other systems, such as the taking a user's gaze from an AR/VR headset to select an object

Video understanding

A video is a **sequence** of images

4D tensor: T x 3 x H x W

(or 3 x T x H x W)

Video classification

Images: Recognize **objects**

Dog
Cat
Fish
Truck

Videos: Recognize **actions**

Swimming
Running
Jumping
Eating
Standing

Problem: videos are big!

Input video:
T x 3 x H x W

Videos are ~30 frames per second (fps)

Size of uncompressed video
(3 bytes per pixel):

SD (640 x 480): **~1.5 GB per minute**
HD (1920 x 1080): **~10 GB per minute**

Problem: videos are big!

Input video:
T x 3 x H x W

Videos are ~30 frames per second (fps)

Size of uncompressed video
(3 bytes per pixel):

SD (640 x 480): **~1.5 GB per minute**
HD (1920 x 1080): **~10 GB per minute**

Solution: Train on short **clips:** low fps and low spatial resolution
e.g. T = 16, H=W=112
(3.2 seconds at 5 fps, 588 KB)

Training on clips

Raw video: Long, high FPS

Training on clips

Raw video: Long, high FPS

Training: Train model to classify short **clips** with low FPS

Training on clips

Raw video: Long, high FPS

Training: Train model to classify short **clips** with low FPS

Testing: Run model on different clips, average predictions

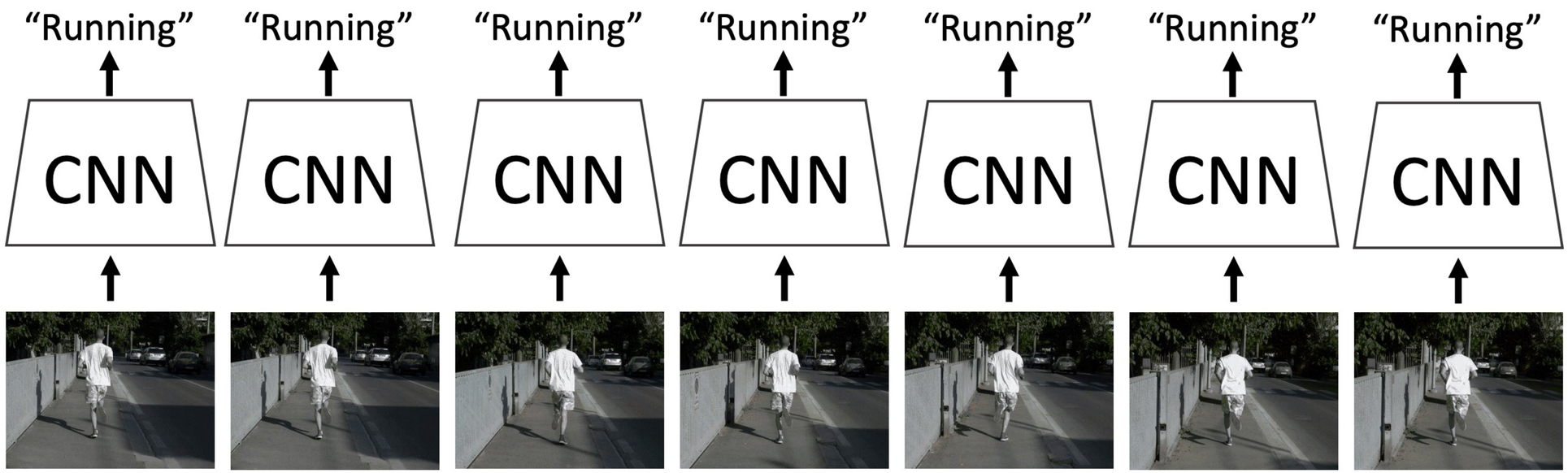
Video classification: single-frame CNN

Simple idea: train normal 2D CNN to classify video frames independently!

Average predicted probabilities at test time.

Often a very strong baseline for video classification.

Advantage: can reuse “well-understood” image architectures.

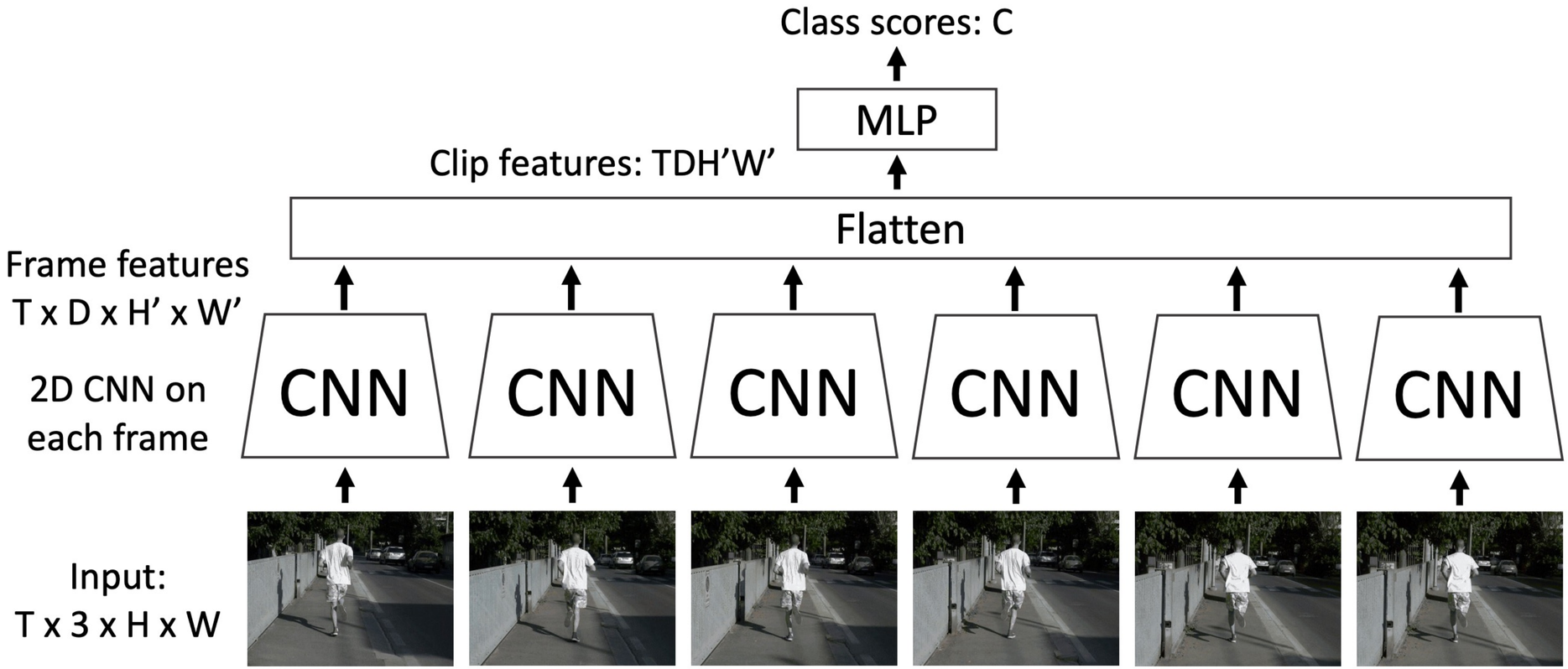


Video classification: Late fusion

Intuition: Get high-level appearance of each frame, and combine them.

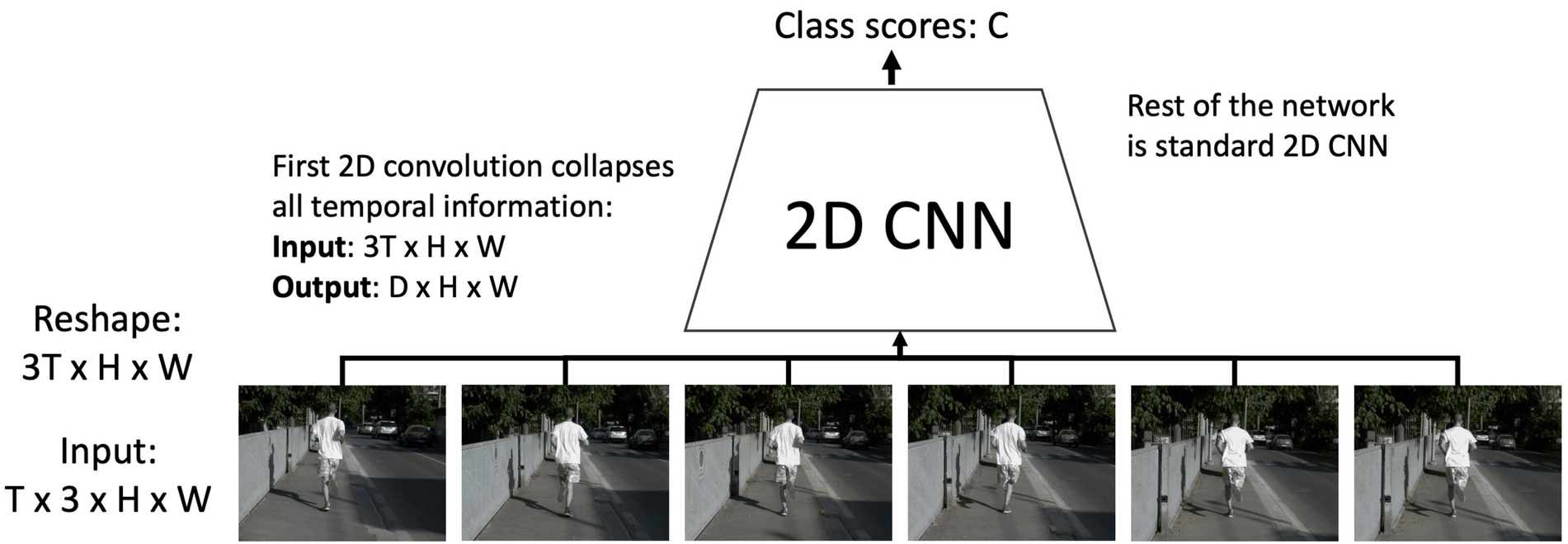
Run 2D CNN on each frame, concatenate features and feed to MLP.

Advantage: can use feature encoders that are pre-trained on images.



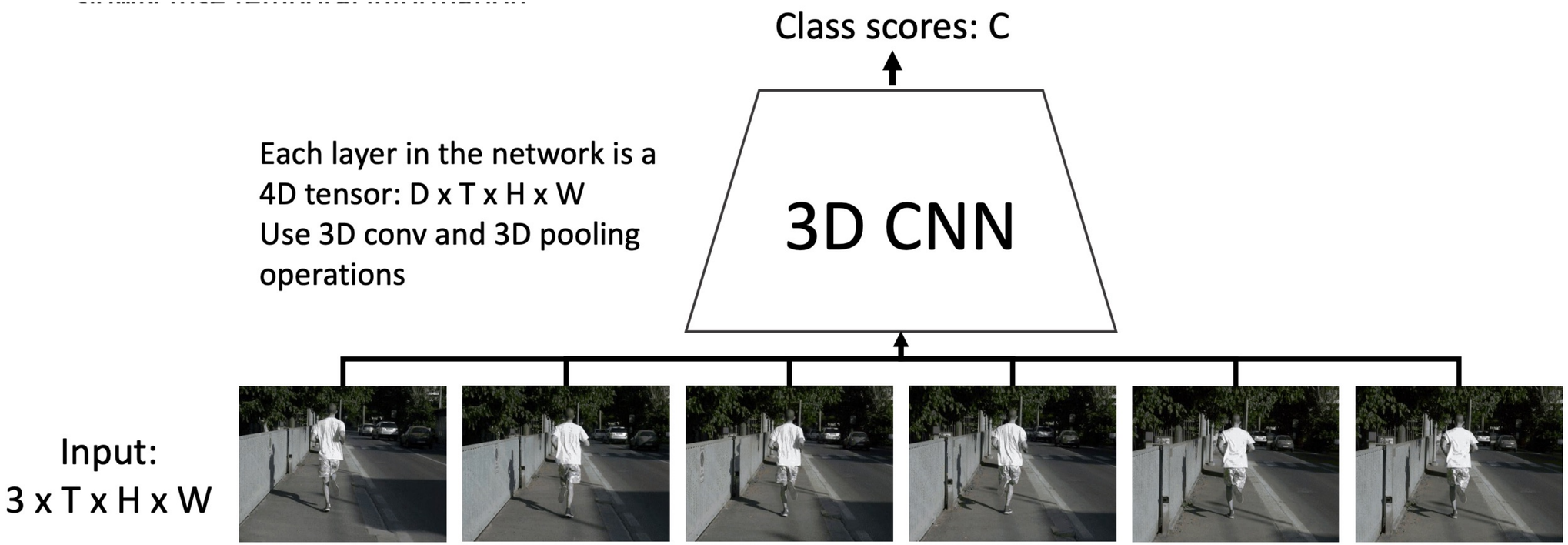
Video classification: 2D CNN

Intuition: Compare frames with very first conv layer, after that normal 2D CNN.



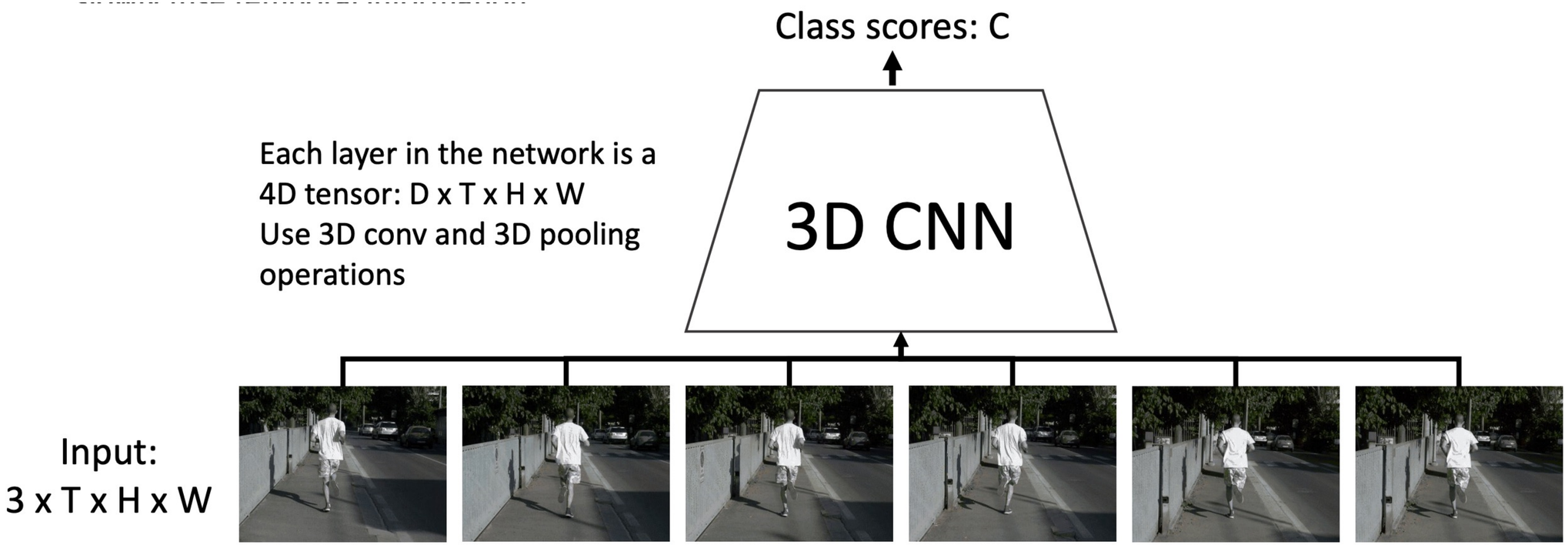
Video classification: 3D CNN

Intuition: Use 3D versions of convolution and pooling to slowly fuse temporal information over the course of the network



Video classification: 3D CNN

Intuition: Use 3D versions of convolution and pooling to slowly fuse temporal information over the course of the network



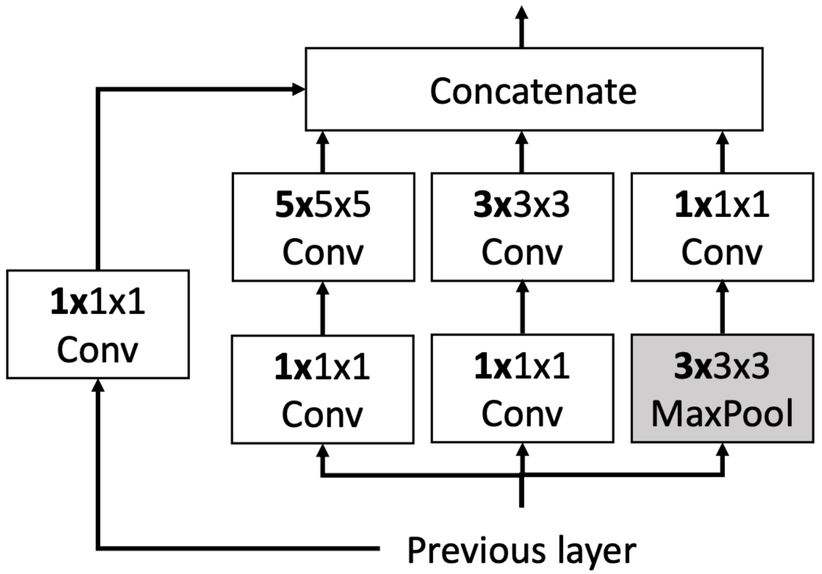
Inflating 2D networks to 3D

Intuition: there has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D K_h x K_w conv/pool layer with a 3D K_t x K_h x K_w version

Inception Block: Inflated



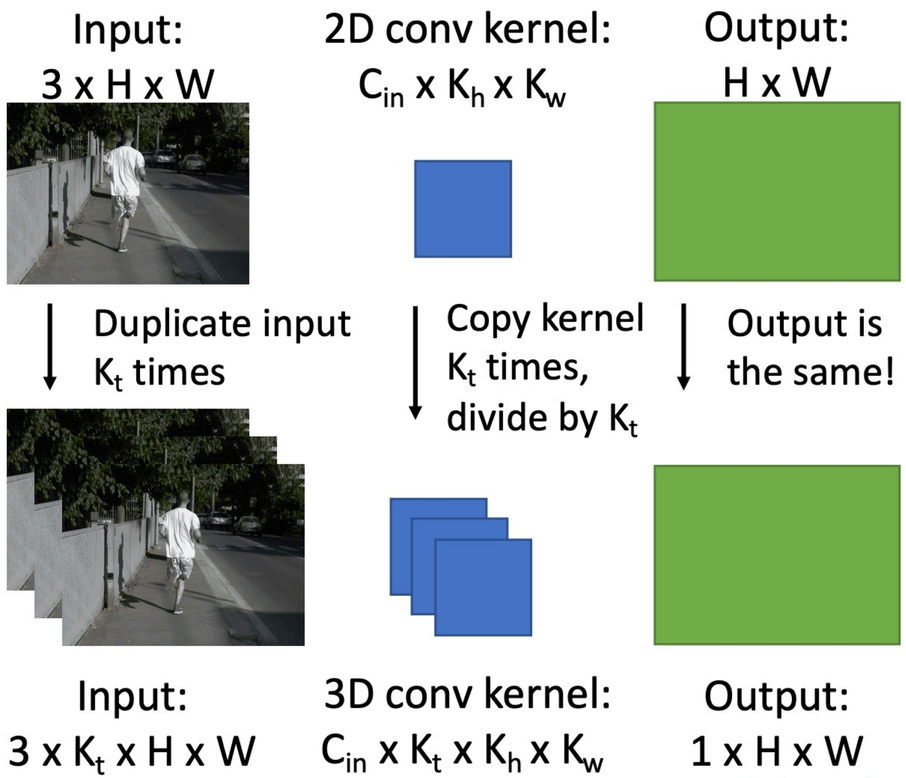
Inflating 2D networks to 3D

Intuition: there has been a lot of work on architectures for images.
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

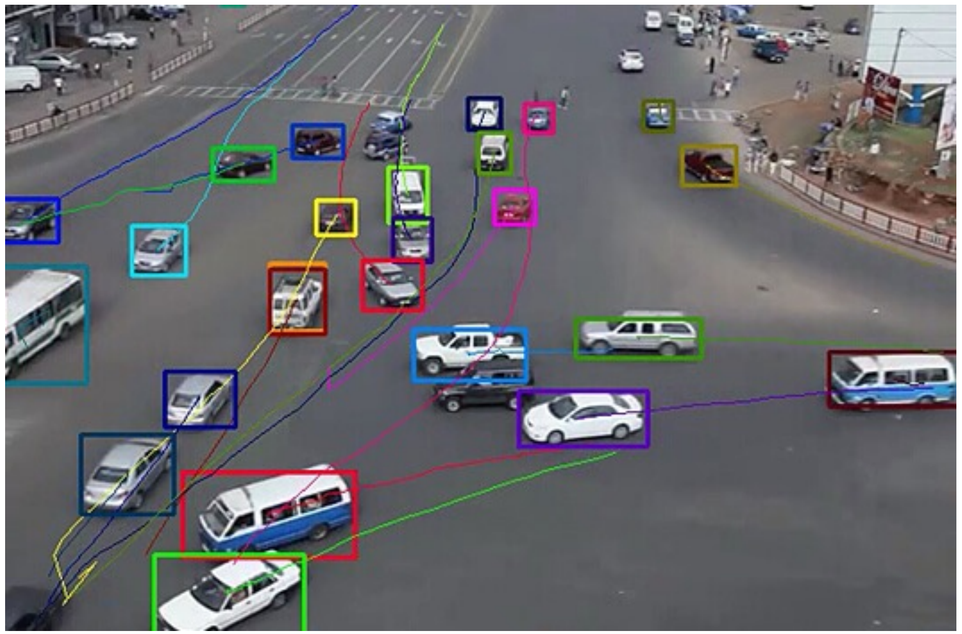
Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

Can use weights of 2D conv to initialize 3D conv: copy K_t times in space and divide by K_t
This gives the same result as 2D conv given "constant" video input



Spatio-temporal detection and tracking

Task: Detect objects of interest in frames and track them over time



Lab 3 this afternoon!

Spatio-temporal detection and tracking

Task: Detect objects of interest in frames and track them over time

frame 1

...

frame N

Lab 3 this afternoon!

Today's focus

- Semantic segmentation
- Extending R-CNN to other tasks
- Instance segmentation
- Panoptic segmentation
- Human pose estimation
- .. and others

Today's focus

- Semantic segmentation
- Extending R-CNN to other tasks
- Instance segmentation
- Panoptic segmentation
- Human pose estimation
- .. and others

Tomorrow: generative models!

Questions?