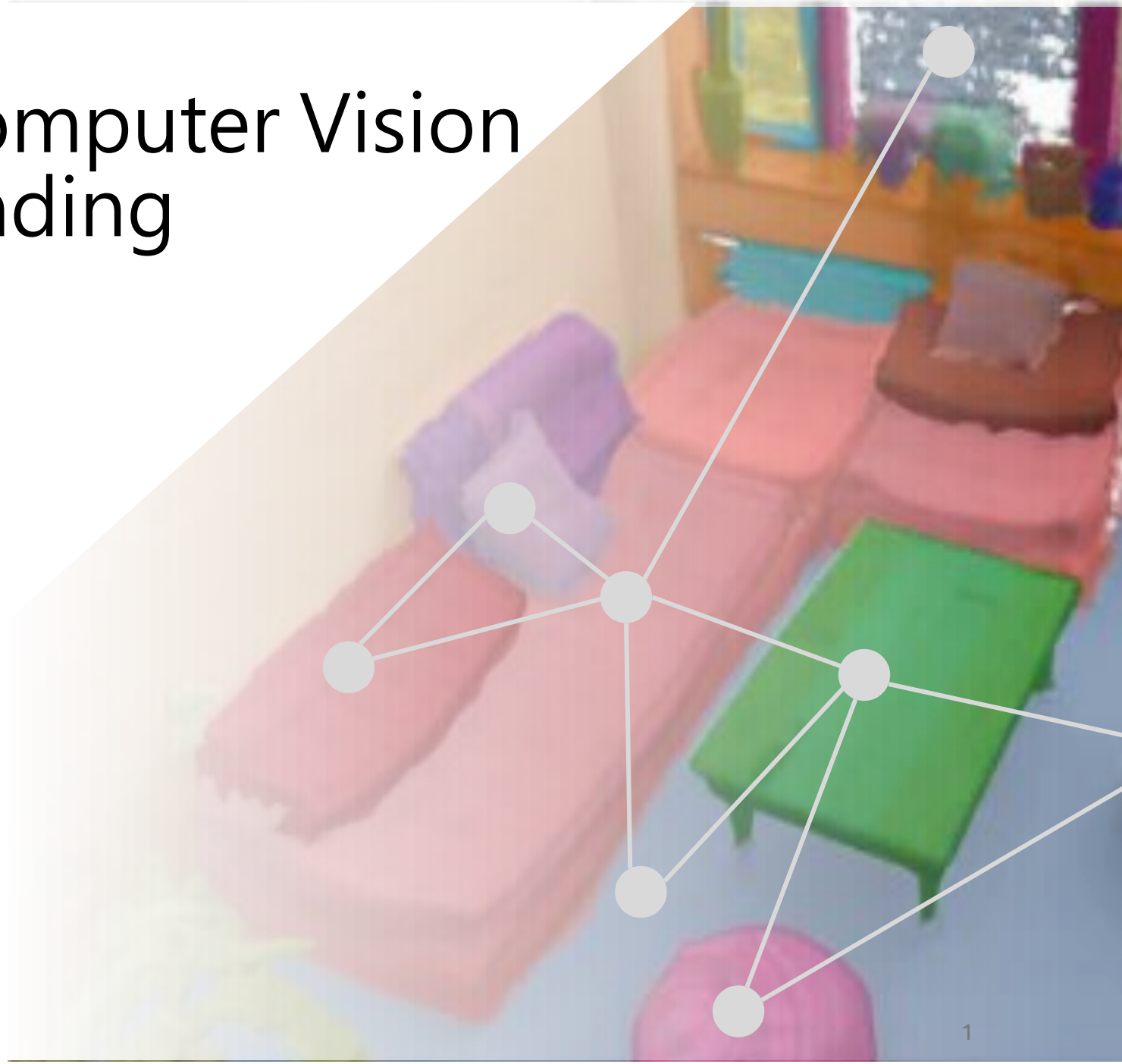
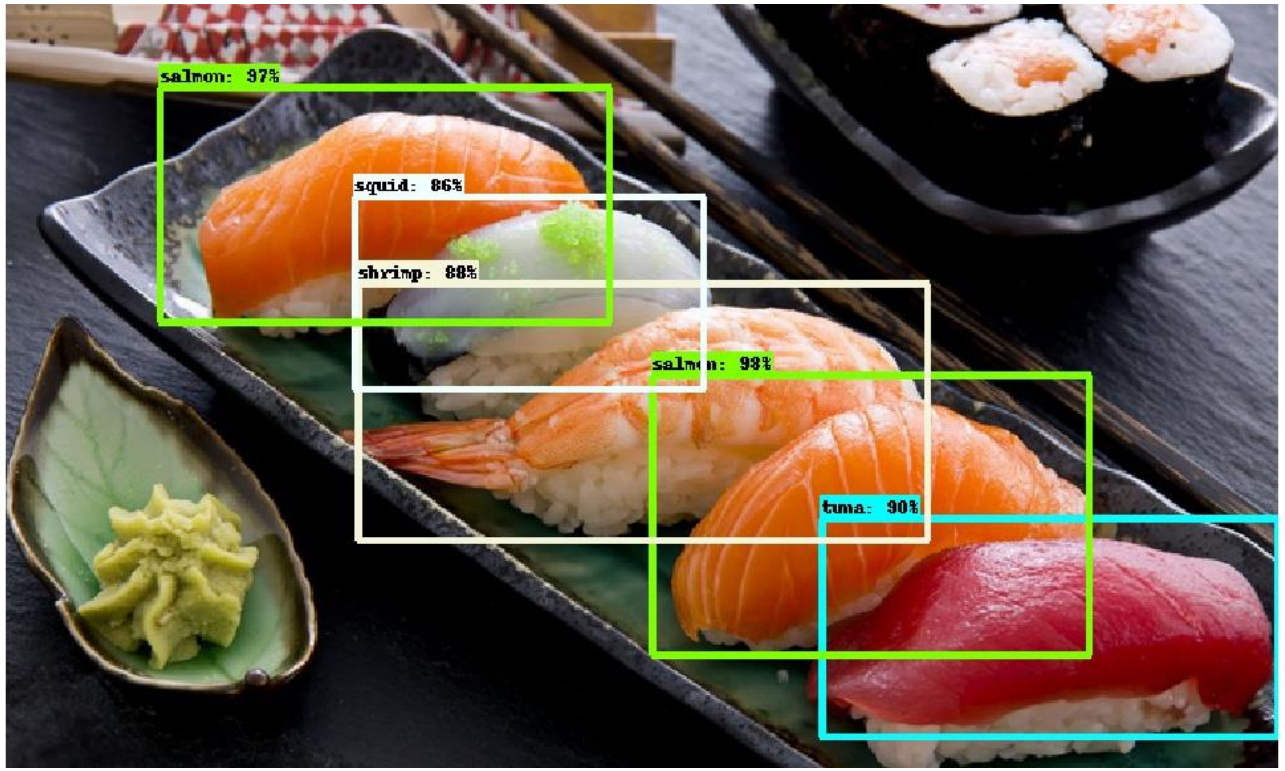
Deep Learning for Computer Vision and Scene Understanding



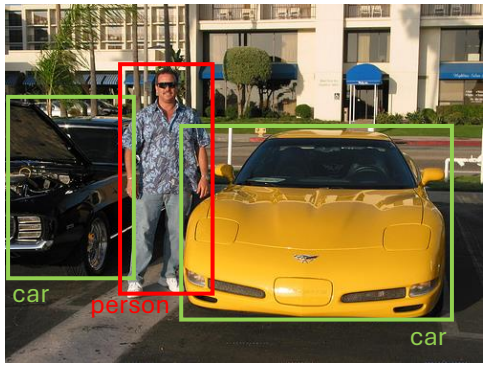
Dr. Helisa Dhamo
Research Scientist in 3D Vision
Huawei Noah's Ark Lab London

Object Detection



Object Detection

- From multi-class classification to multiple instances



- It involves object recognition (what is it?) and localization (where is it?)
- How is it different to classification?
  - Predict the object category (classification) and its bounding box (regression of location and spatial extend of the object)
  - Multiple objects can exist in an image
  - Multiple scales (also one of the biggest challenges)

R-CNN

Regions with CNN features

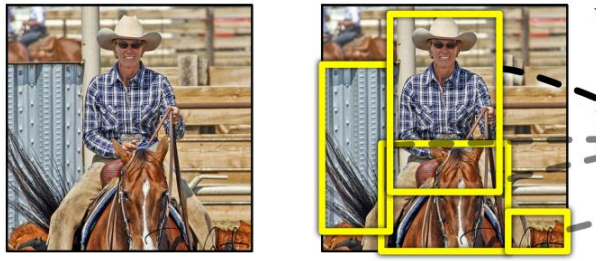
Input image

pipeline

Girshick et al., “Rich feature hierarchies for accurate object detection and semantic segmentation.” CVPR’14

R-CNN

Regions with CNN features

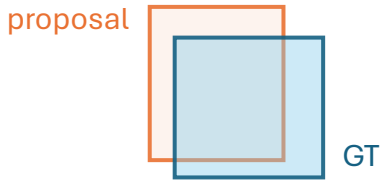


Input image

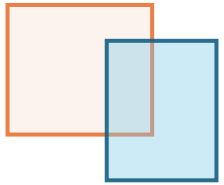
Region proposals

pipeline

- **Region proposals:** Generate class-independent candidate boxes with high recall
- Methods: Selective search, EdgeBox, etc.
- Intersection over Union (IoU) with the ground truth:



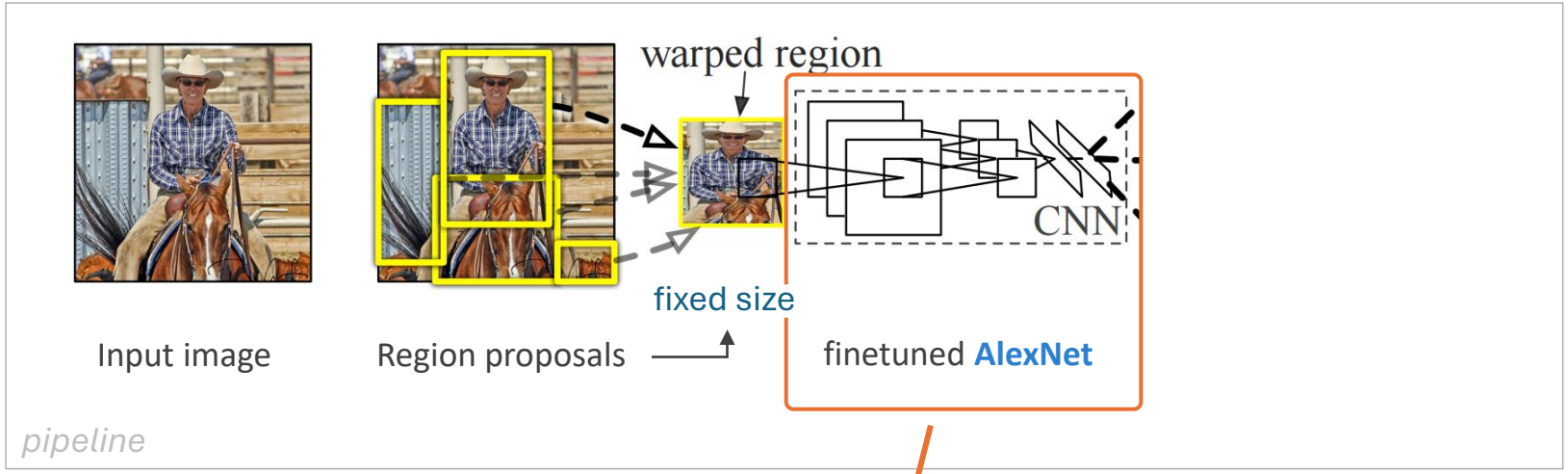
IoU ≥ 0.5:
proposal box is a **positive** candidate for GT class



IoU < 0.5:
proposal box is a **negative** example (background)

R-CNN

Regions with CNN features



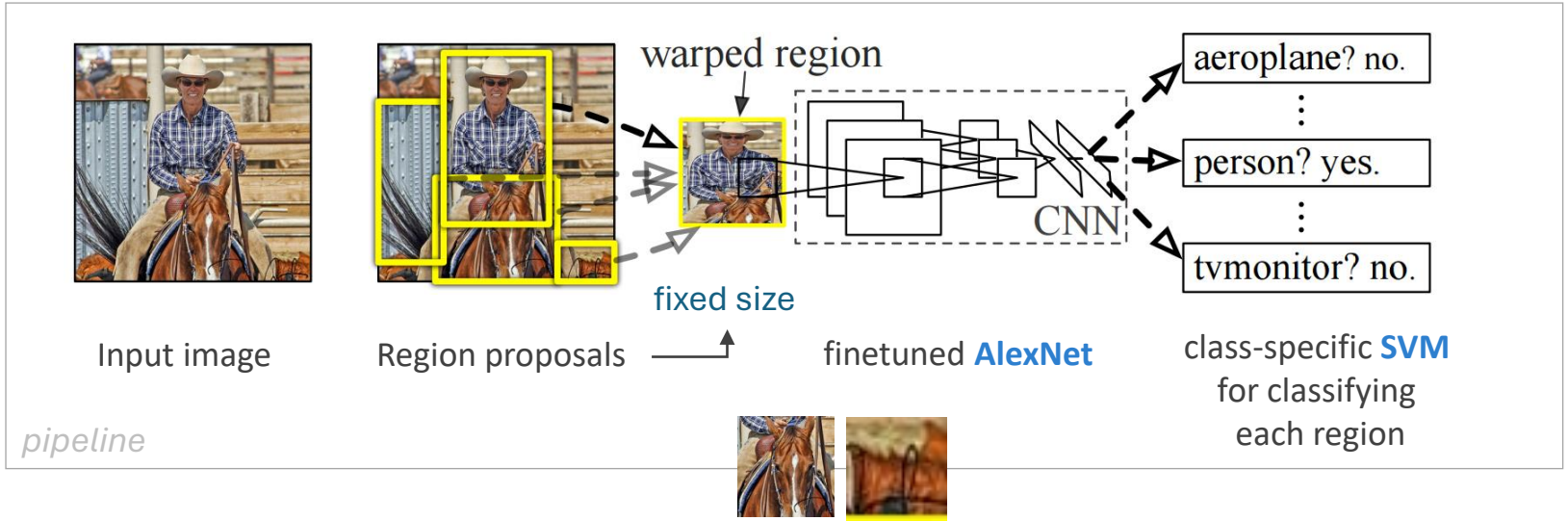
- Warp each region proposal into fixed size
- Training:
  - Fine-tune classification CNN on Pascal VOC dataset
  - Replace 1000-d classification layer with a 21-d (20 classes + background)
- Testing:
  - Feed extracted regions image samples into the finetuned CNN

Girshick et al., “Rich feature hierarchies for accurate object detection and semantic segmentation.” CVPR’14

R-CNN

Regions with CNN features

47s / image

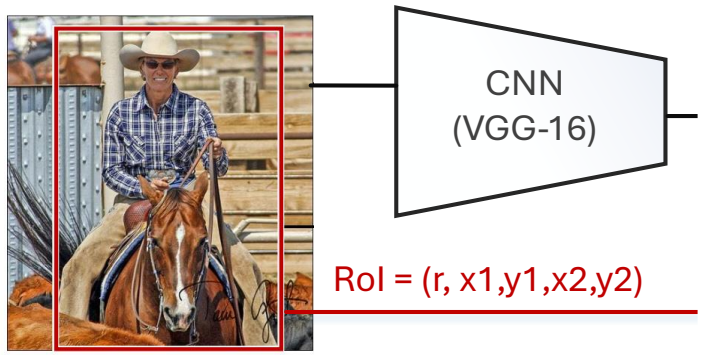


- CNN fully-connected layer as fixed-size feature vector (4096-d)
- Train a linear Support Vector Machine (SVM) *per class* as detection classifier
- Non-maximum suppression based on all regions' scores

Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation." CVPR'14

Fast R-CNN

Classify the region proposals inside the CNN

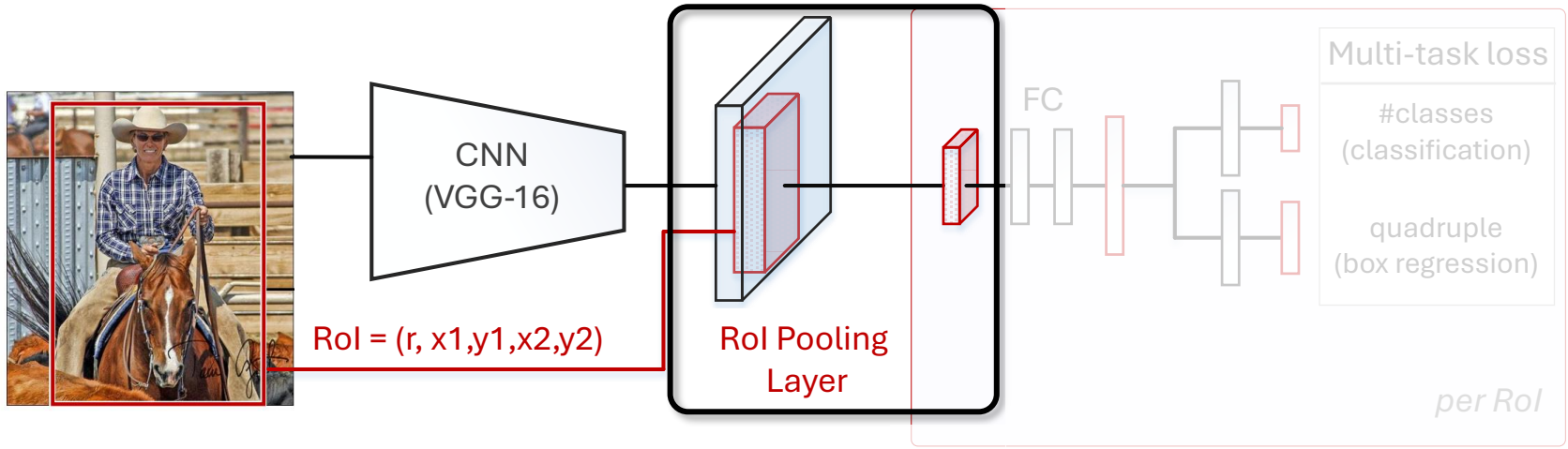


- Room for improvement:
  Share computation of CNN features for all images regions
  (i.e. feed the whole image into the CNN, instead of each region)

Girshick “Fast r-cnn.” ICCV’15.

Fast R-CNN

Classify the region proposals inside the CNN

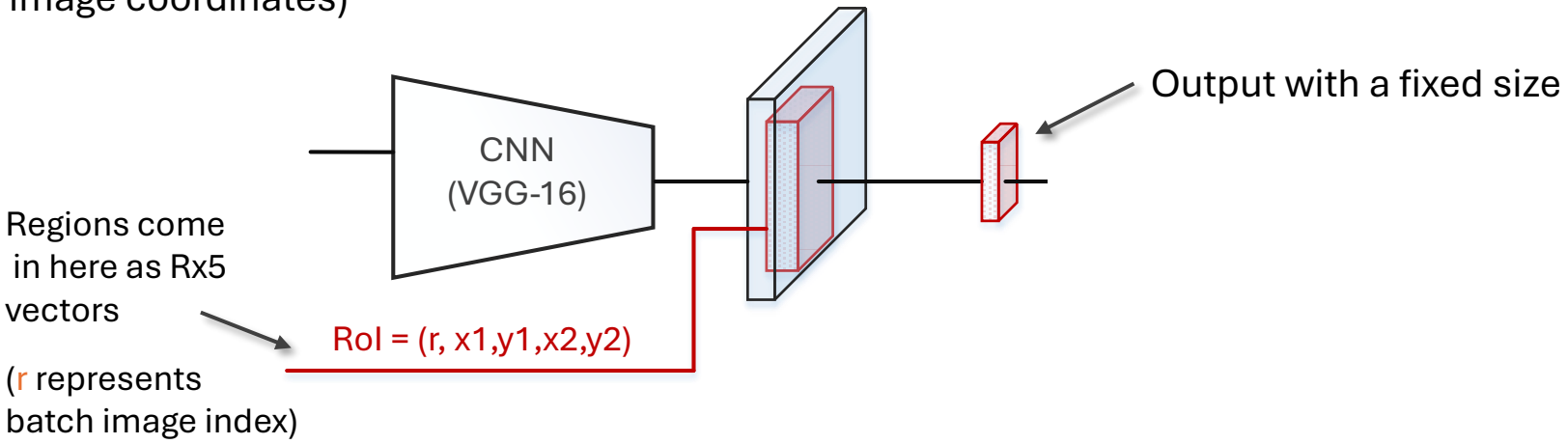


- Replace a max-pooling layer (in the paper: `pool5`) with a custom RoI Pooling Layer
- It handles the regions, after most of the computation has been shared

Girshick “Fast r-cnn.” ICCV’15.

RoI Pooling Layer

- Takes as input a set of feature maps and a set of regions R (defined on image coordinates)

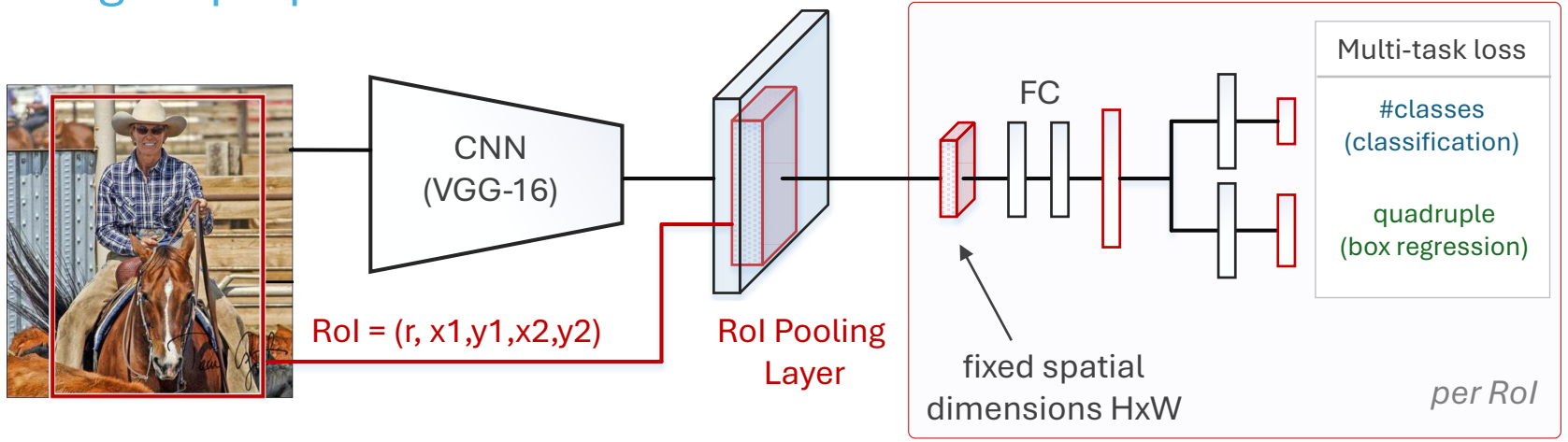


- Project each RoI onto the feature maps
- Divide the arbitrarily sized RoI window $h \times w$, into a **fixed** grid $H \times W$
- Perform max-pooling in each grid cell of approx. size $\frac{h}{H} \times \frac{w}{W}$
- The output size should match the following FC layer (e.g. for VGG $H = 7, W = 7$)
- Effectively changes the batch size from *#images* to *#RoIs*

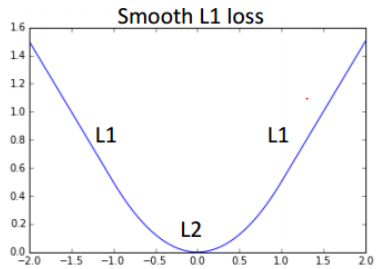
Girshick "Fast r-cnn." ICCV'15.

Fast R-CNN

Classify the region proposals inside the CNN



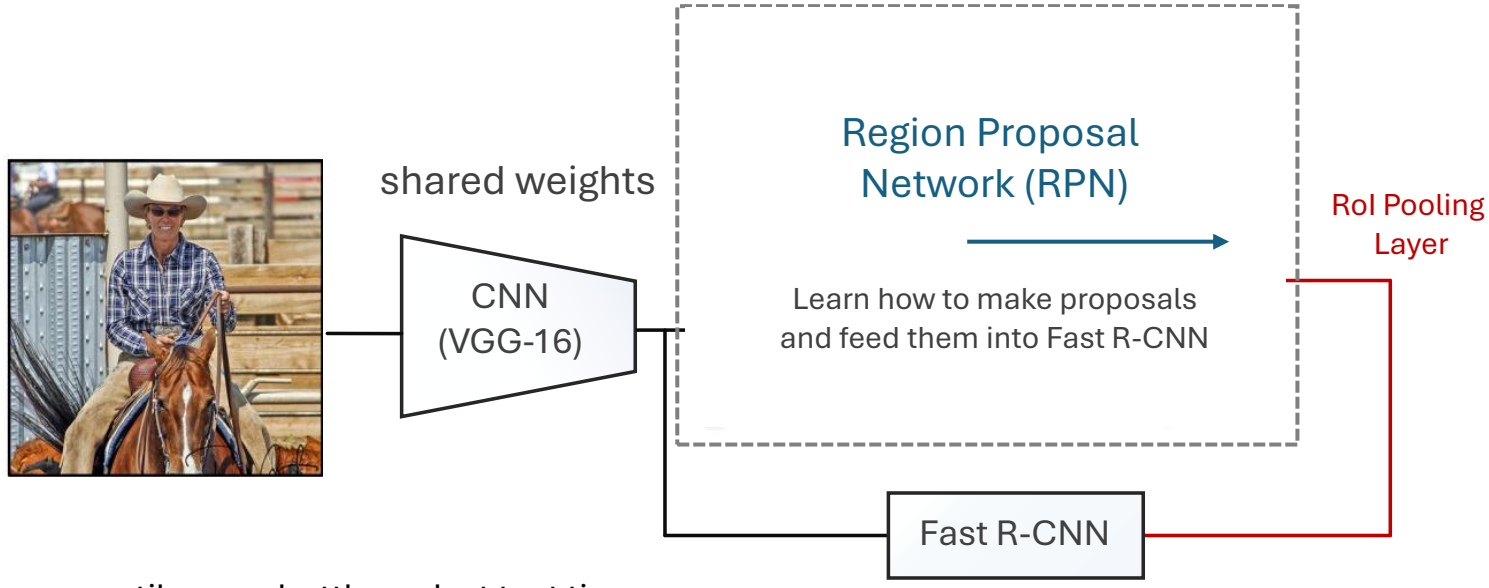
- Follow with FC layers from the original network (e.g. VGG)
- Compute an output for each RoI independently
- Classify the RoI (#RoIs × 1 × #classes)
  - standard cross entropy
- Refine the box location *per class* (#RoIs × 4 × #classes)
  - Smooth L1 loss



Girshick "Fast r-cnn." ICCV'15.

Faster R-CNN

Unify object detection CNN and region proposal generation

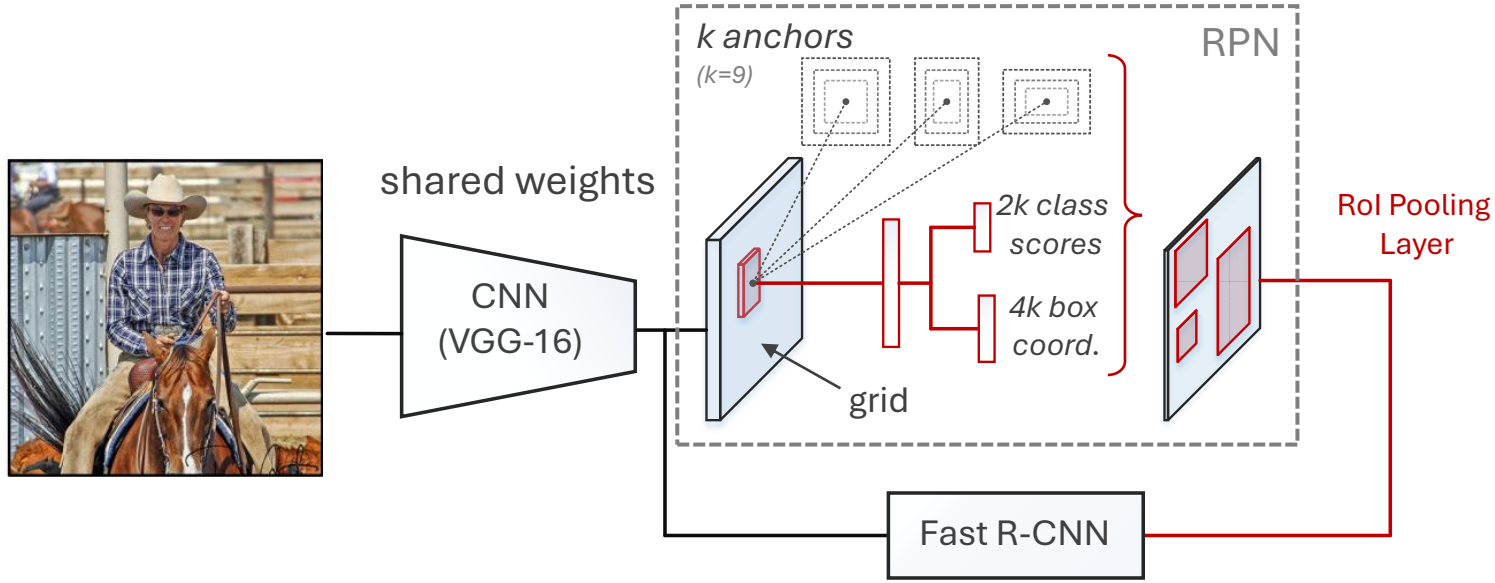


- Proposal generation was until now a bottleneck at test time
- First, keep a core computation branch (VGG) with shared weights
- Then, split into two subparts:
  - Proposal generator (RPN)
  - Object detector (Fast R-CNN)

Ren et al., “Faster r-cnn: Towards real-time object detection with region proposal networks.” *NIPS’15*

Faster R-CNN

Unify object detection CNN and region proposal generation

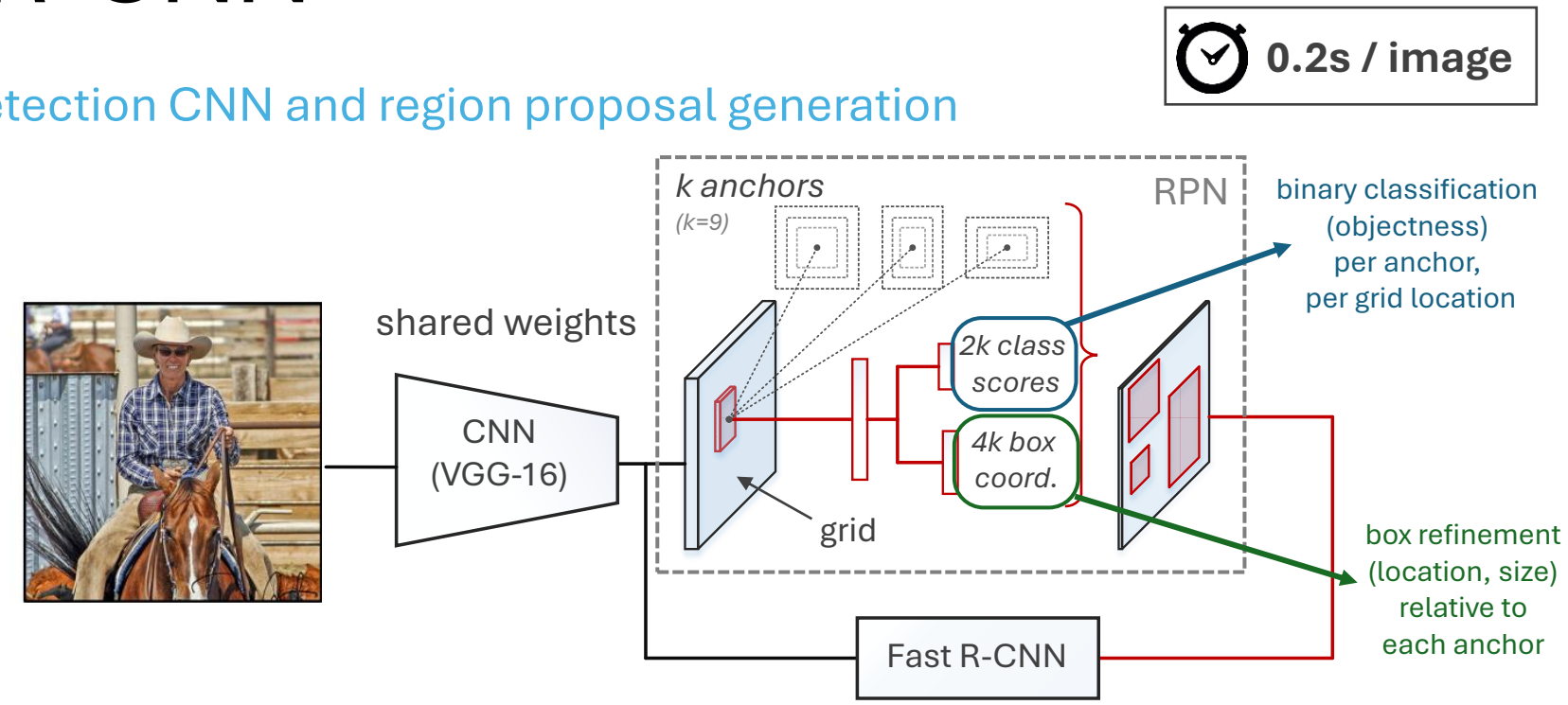


- RPN tells Fast R-CNN where to look
- Learn to generate class agnostic proposals by regressing region bounds and object-ness scores with respect to reference anchors at *each* spatial location
- Object-ness = confidence of region belonging to *any* class vs. background

Ren et al., "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS'15*

Faster R-CNN

Unify object detection CNN and region proposal generation



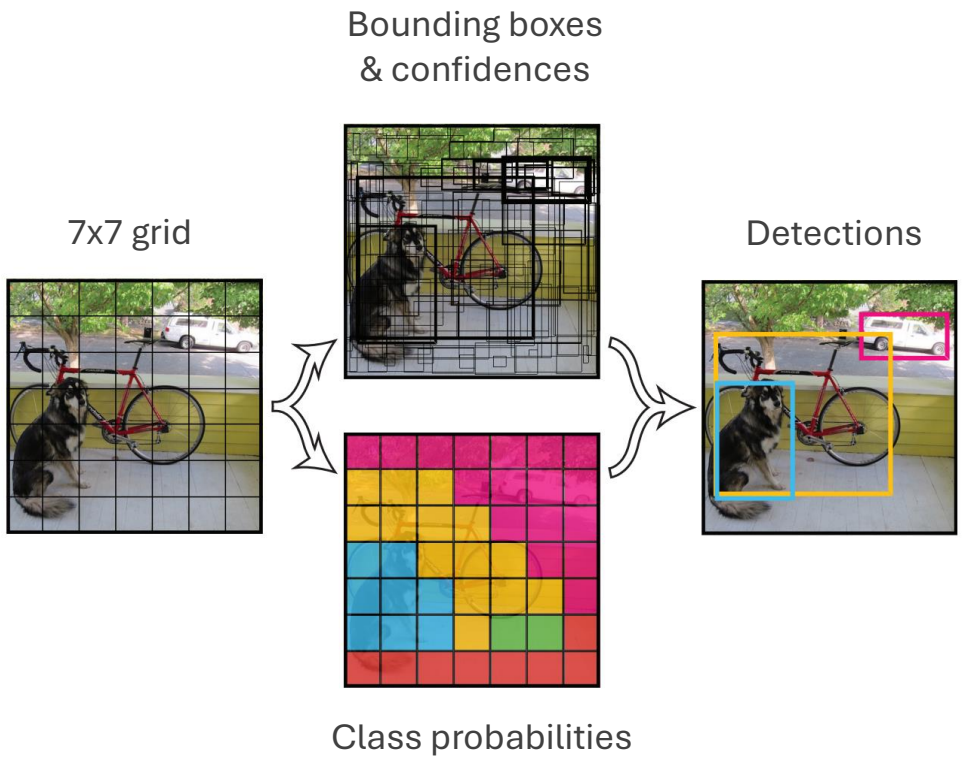
- Anchors represent "default" boxes of different sizes and aspect, placed on the grid
  - sizes 128^2, 256^2, 512^2, aspect ratios 1:1, 1:2, 2:1
- Maximum k proposals per grid cell ($k = 9$ for this configuration)
- Each grid cell → feature vector (i.e. fully connected layers in a sliding window)
- Multi-task prediction (two losses for RPN + two losses for Fast R-CNN)

Ren et al., "Faster r-cnn: Towards real-time object detection with region proposal networks." *NIPS'15*

YOLO: You Only Look Once

Detection without proposals

- Divide image into fixed grid (7×7)
- Predict for each grid cell:
  - Bounding boxes (x, y, w, h) with the center of the box relative to cell (total: 98 detections per class)
  - "Objectness" confidences
  - Class-specific confidences
- Standard architecture (e.g. VGG)
- Reshaped fully connected layer to match the grid



Redmon et al., "You only look once: Unified, real-time object detection." CVPR'16

YOLO: You Only Look Once

Unify object detection components into a single step

< 0.05s / image

Performance:
66.4% (VGG)

Limitations:
Each cell predicts only 2 boxes and one class, thus only 98 possible detections in the entire image.

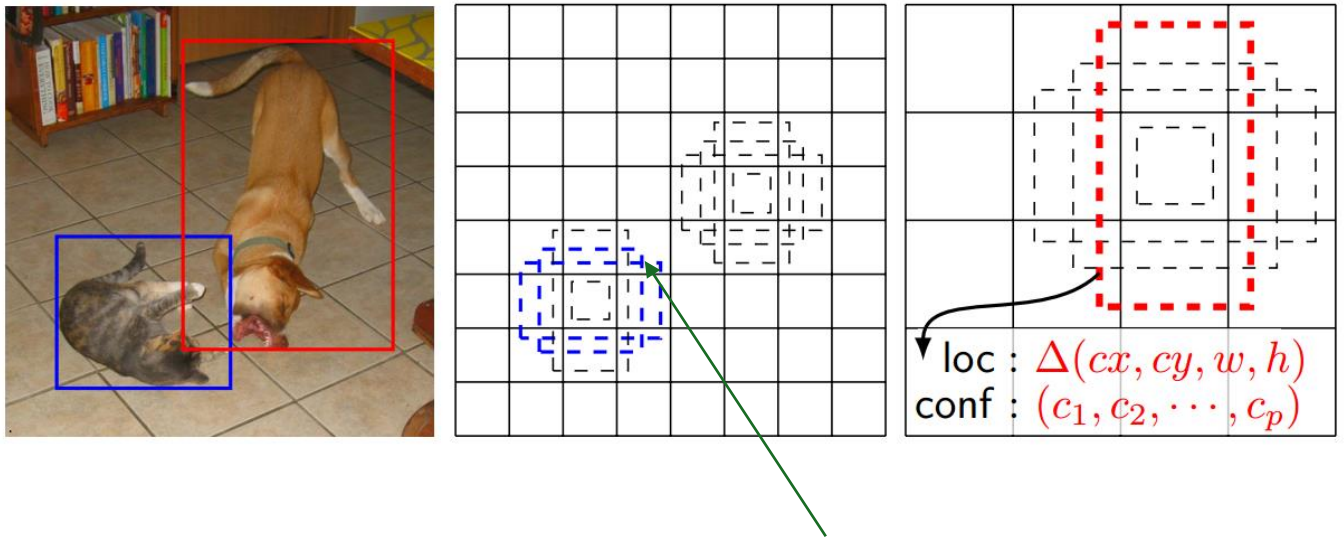
Difficulty with small, grouped objects due to this spatial coarseness.

Much faster but less accurate than Faster R-CNN

SSD: Single Shot Detector

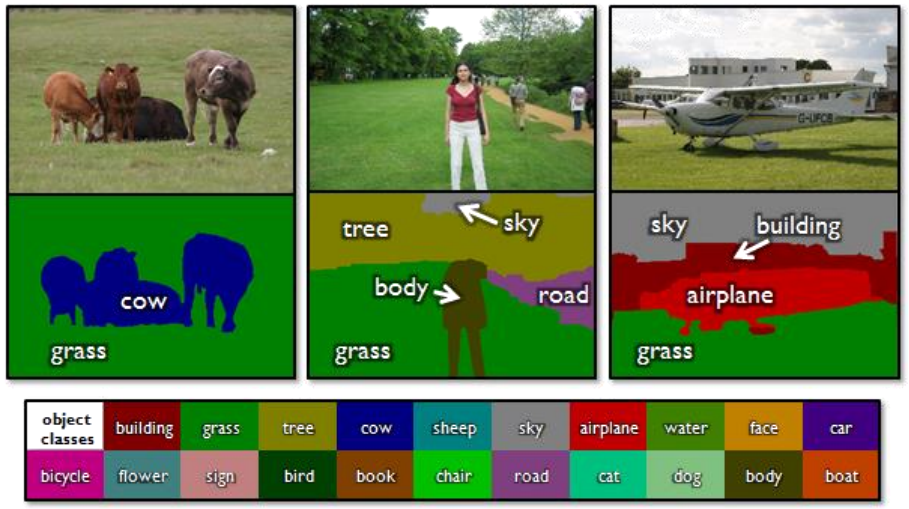
Grid-based object detection with anchors

- Divide image into grid space, similar to YOLO



- Reference bounding boxes at each cell, similar to RPN anchors

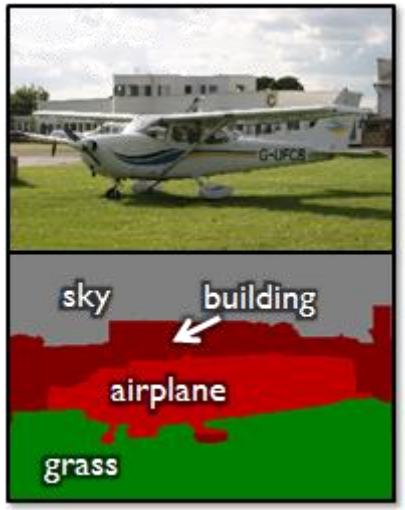
Semantic Segmentation



Semantic Segmentation

Per-pixel labelling of semantic categories

- Classification on *pixel-level* is semantic segmentation (assign every pixel to a class)
- Classification on image level benefits from deep CNNs thanks to their invariance to local transformations (pooling, stride).
- However, in **dense** predictions problems:
  - Loss of spatial information is undesired.
  - The output needs to be a high resolution map (ideally same as input)

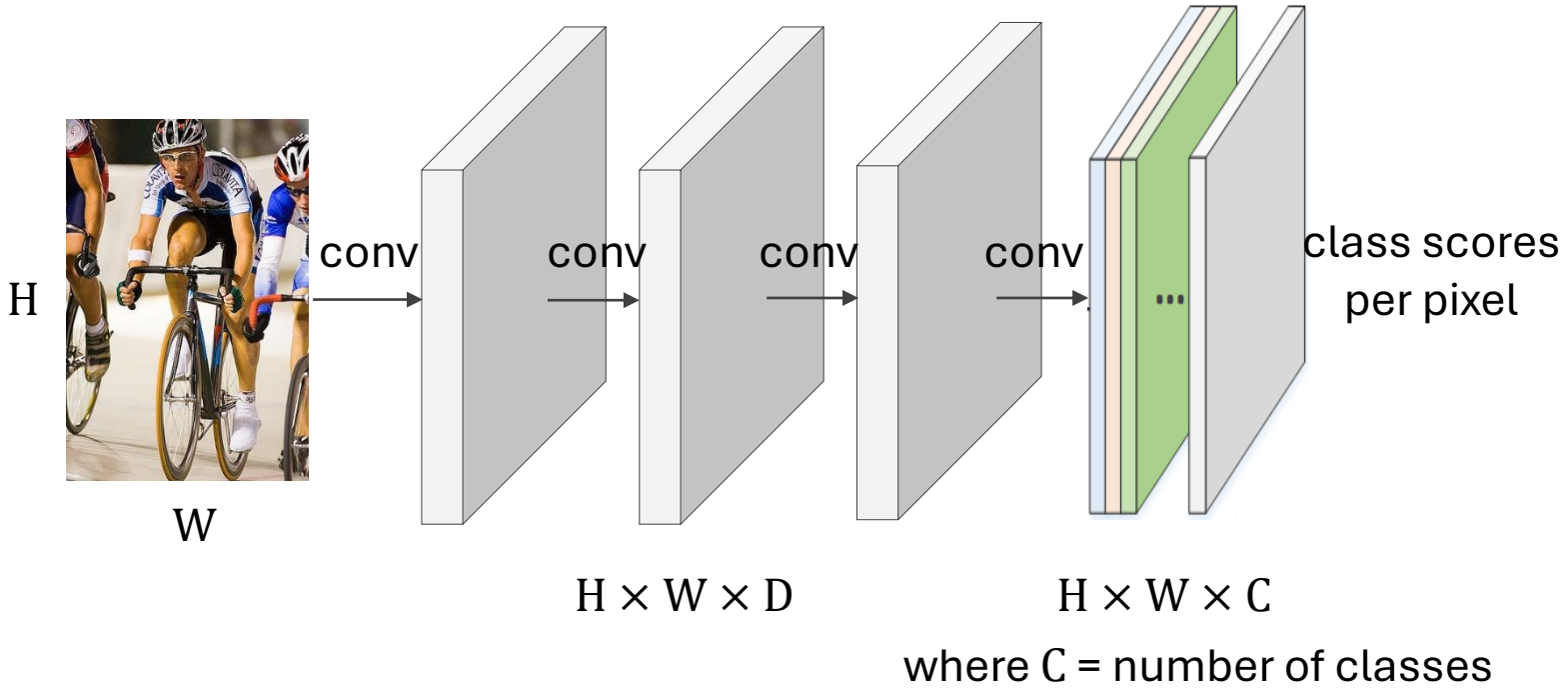


Source: http://jamie.shotton.org/work/research.html

Semantic Segmentation

Only convolutions, stride = 1, i.e. no sub-sampling

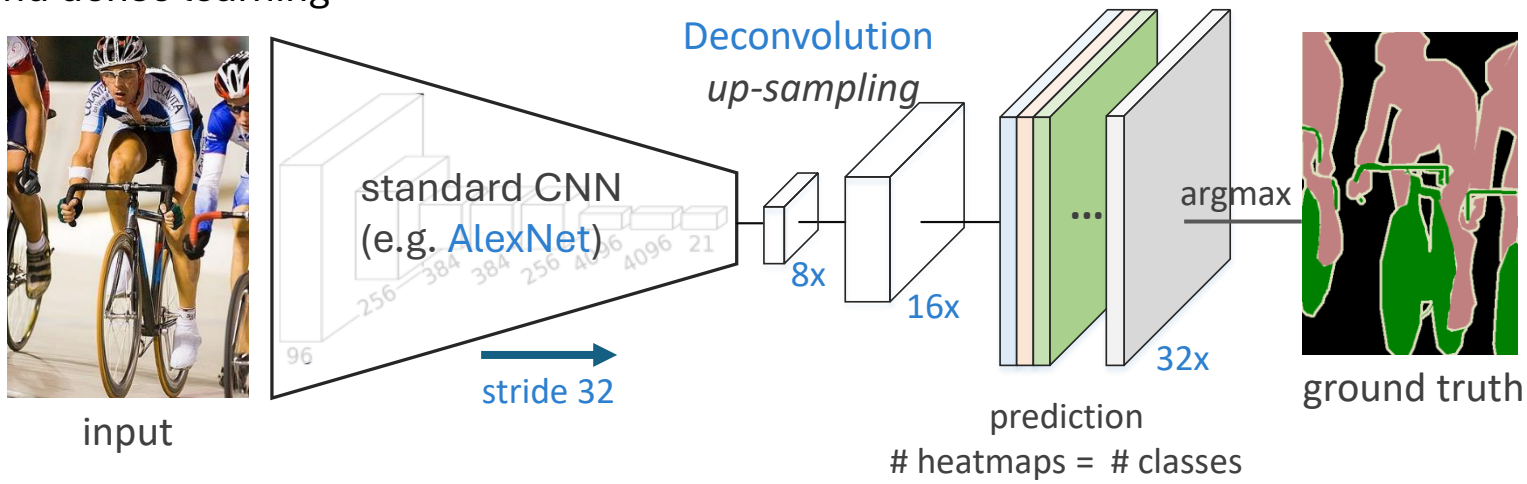
Convolutions at image resolution are *very* expensive



Semantic Segmentation

Fully Convolutional Networks (FCNs)

- Key concept: Replace fully-connected layers with convolutions, such that for larger inputs, the final connection becomes local
- The output then is not a vector, but a heatmap
- Spatially up-sample the low-resolution feature maps (in one-step or gradually)
- Combine coarse (higher layer) with fine (lower layer) information
- End-to-end dense learning

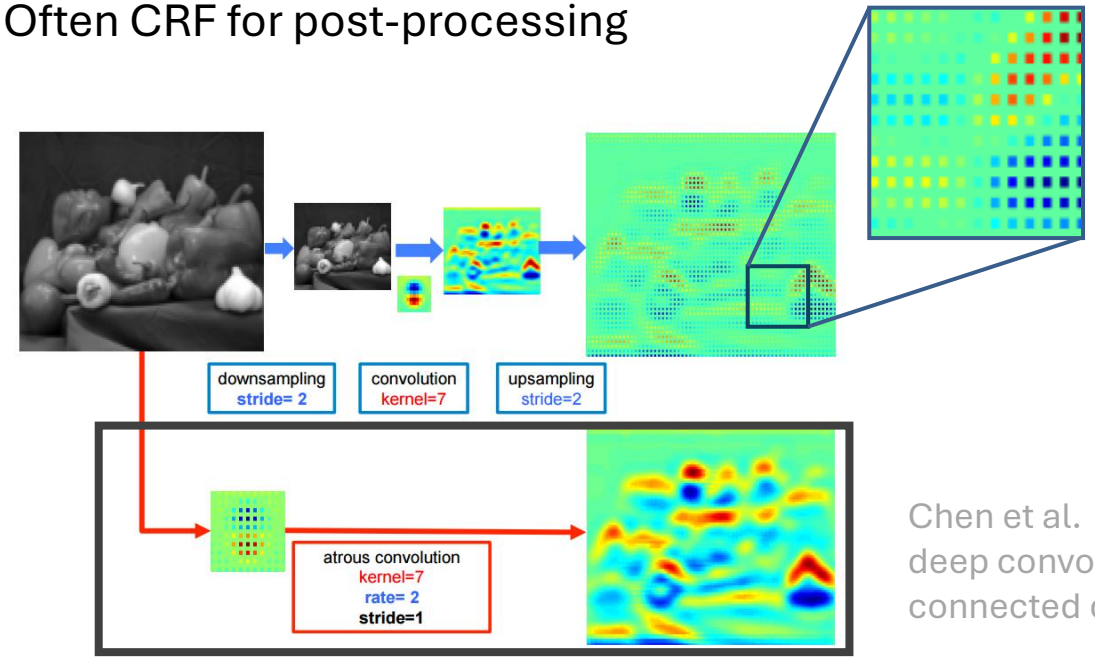


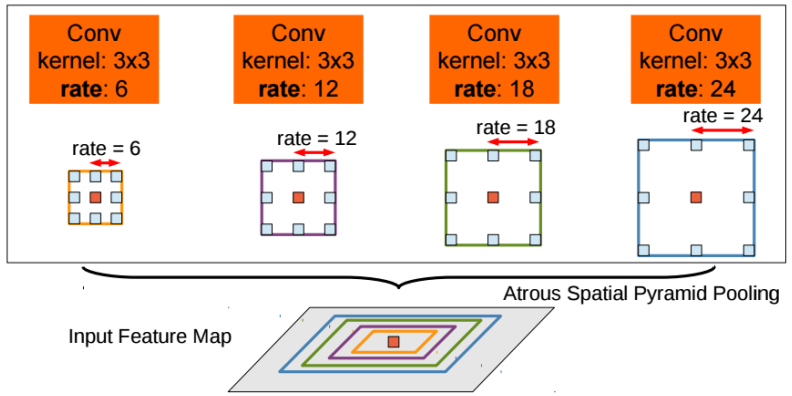
Long et al., "Fully convolutional networks for semantic segmentation." CVPR'15

Semantic Segmentation

DeepLab

- Add sparsity (*holes*) to convolutions or feature maps to *replace pooling* atrous convolutions
- Alternative to deconvolutions (i.e. encoder-decoder style architectures)
- No increase in number of parameters or amount of computation
- Often CRF for post-processing





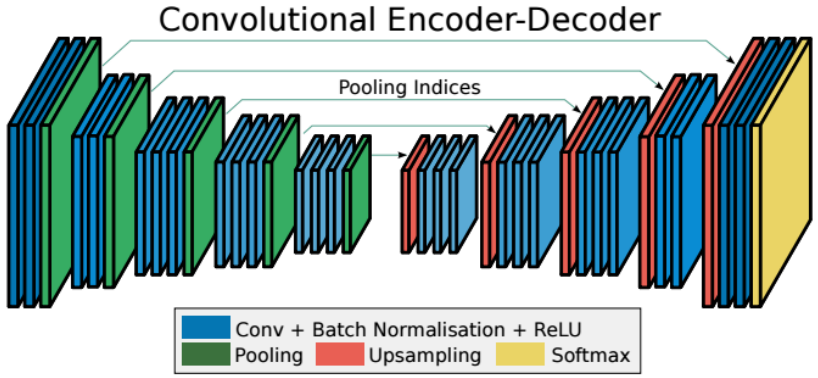
Atrous pyramid

Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *arXiv preprint arXiv:1606.00915* (2016).

Semantic Segmentation

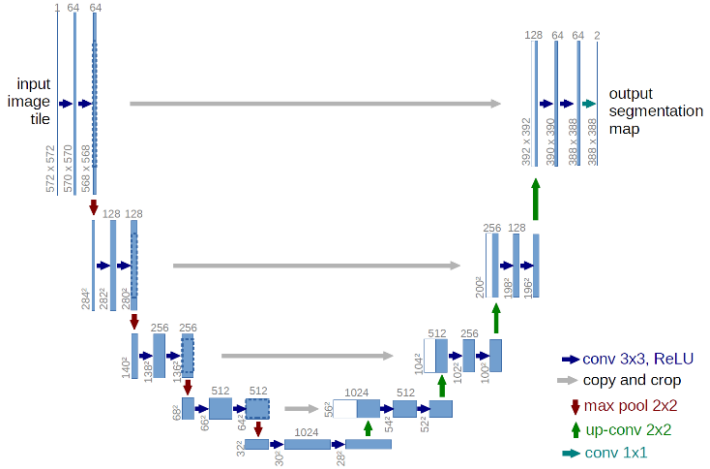
SegNet (road scene segmentation)

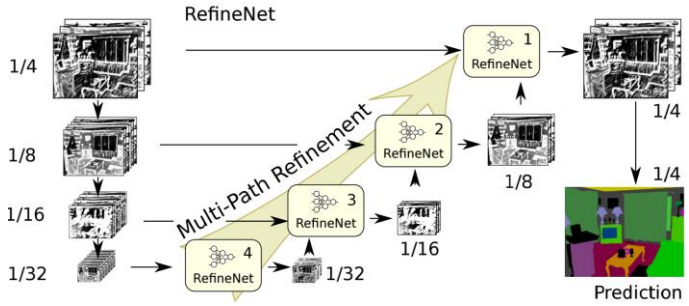
Badrinarayanan et al., “Segnet: A deep convolutional encoder-decoder architecture for image segmentation.” arXiv preprint arXiv:1511.00561. (2015)



U-net (medical image segmentation)

Ronneberger et al., “U-net: Convolutional networks for biomedical image segmentation.” MICCAI’15

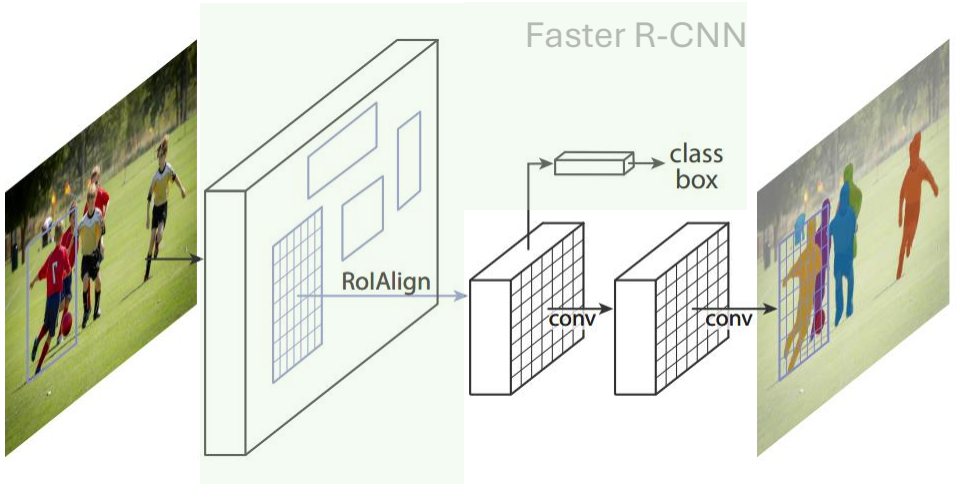




RefineNet (multi-resolution)

Lin et al., “RefineNet: Multi-Path Refinement Networks with Identity Mappings for High-Resolution Semantic Segmentation.” CVPR’17

Instance Segmentation



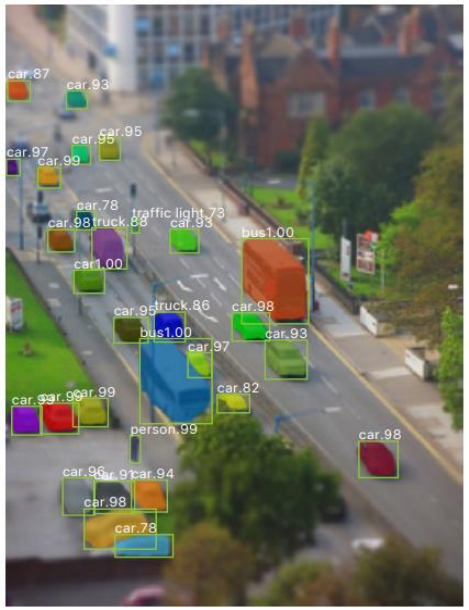
Segment object instances

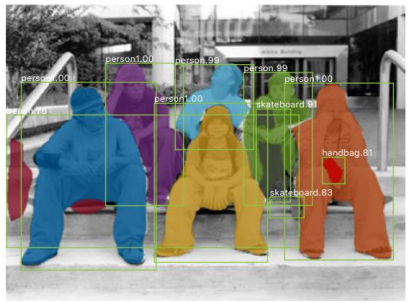
Semantic Segmentation:

Per-pixel labels are semantic classes

Instance Segmentation:

Additionally differentiate among instances of the same class





0.2s / image

- Mask R-CNN
  - Faster R-CNN: Bbox + class per RoI
  - Mask per RoI
  - RoIAlign > RoIPooling

He et al. “Mask R-CNN „ ICCV‘17

Object Detection / Semantic Segmentation: Take home message

- Transfer learning
  (almost) always use classification (ImageNet) architectures as backbone

- Combination of losses
  e.g. bounding box regression + label classification

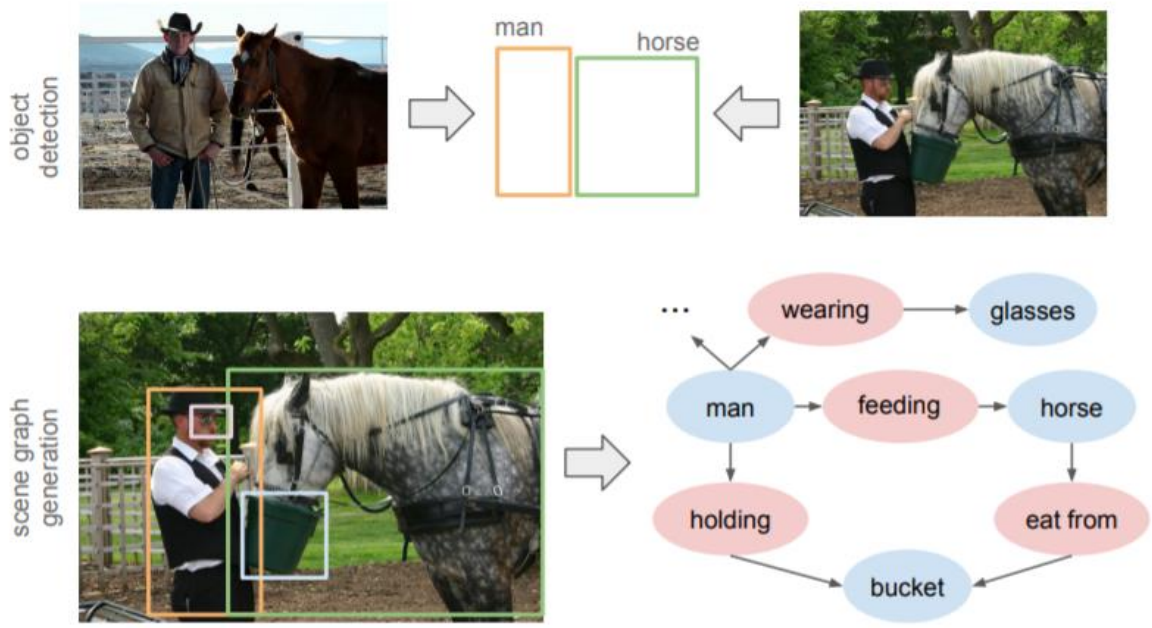
- Different architectures
  e.g. fully convolutional, encoder-decoder

- There is always room for improvement towards a specific task
  e.g. R-CNN → Fast R-CNN → Faster R-CNN …

Scene Understanding Beyond Objects

Scene Graphs

- **Nodes:** objects in the scene
- **Edges:** relationships between objects (interaction, relative position)



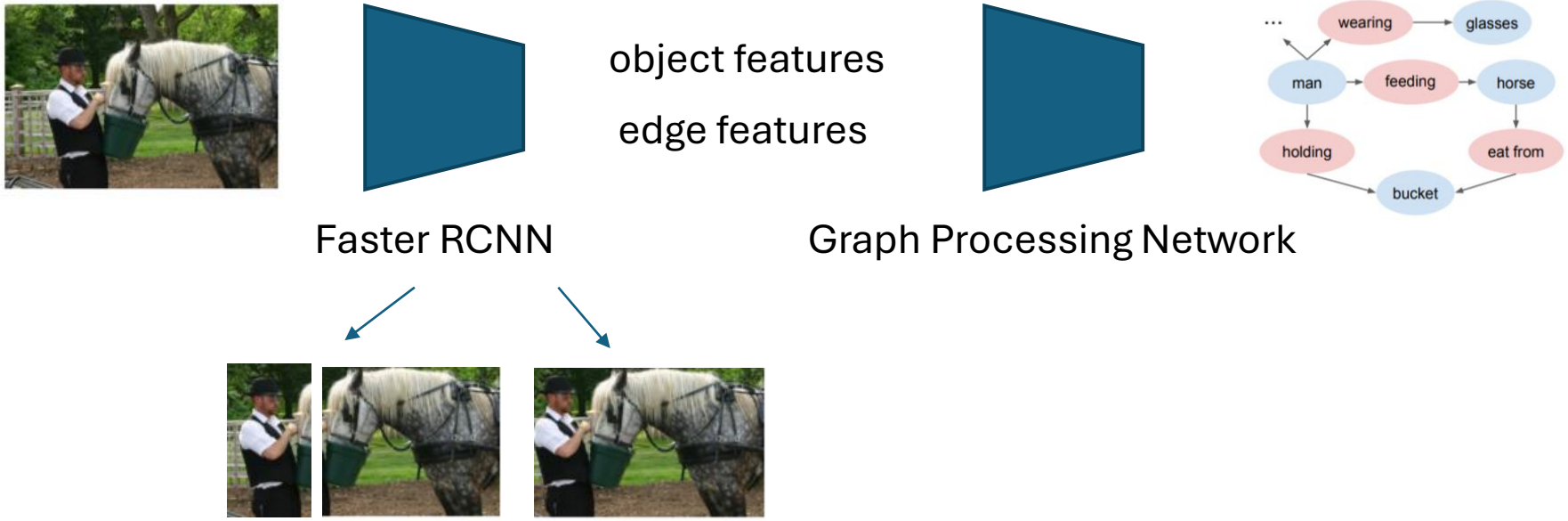
Xu et al., “Scene graph generation by iterative message passing.” CVPR’17

Scene Understanding Beyond Objects

Scene Graphs

- **Nodes:** objects in the scene
- **Edges:** relationships between objects (interaction, relative position)

Scene graph generation networks are usually build on top of an object detector



What I cannot create, I do not understand”

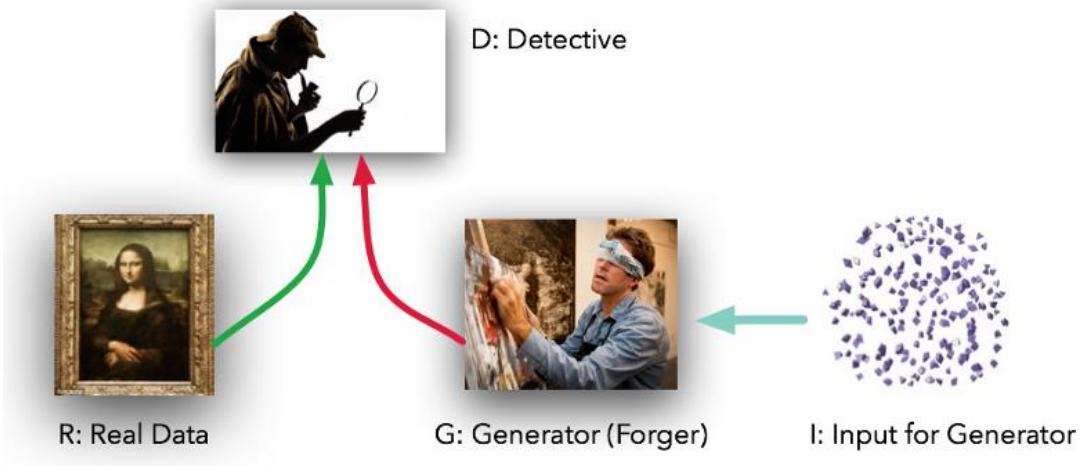
Richard Feynman

Generative Models

Generative Adversarial Networks

Two networks play a two player „game“.

- A generator generates samples from a data distribution.
- A discriminator tries to identify what data is generated and what is real.
- The generator tries to fool the discriminator.



https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

Goodfellow, Ian, et al. “Generative adversarial nets” NIPS 2014.

Generative Models

Pix2Pix : Paired image-to-image translation

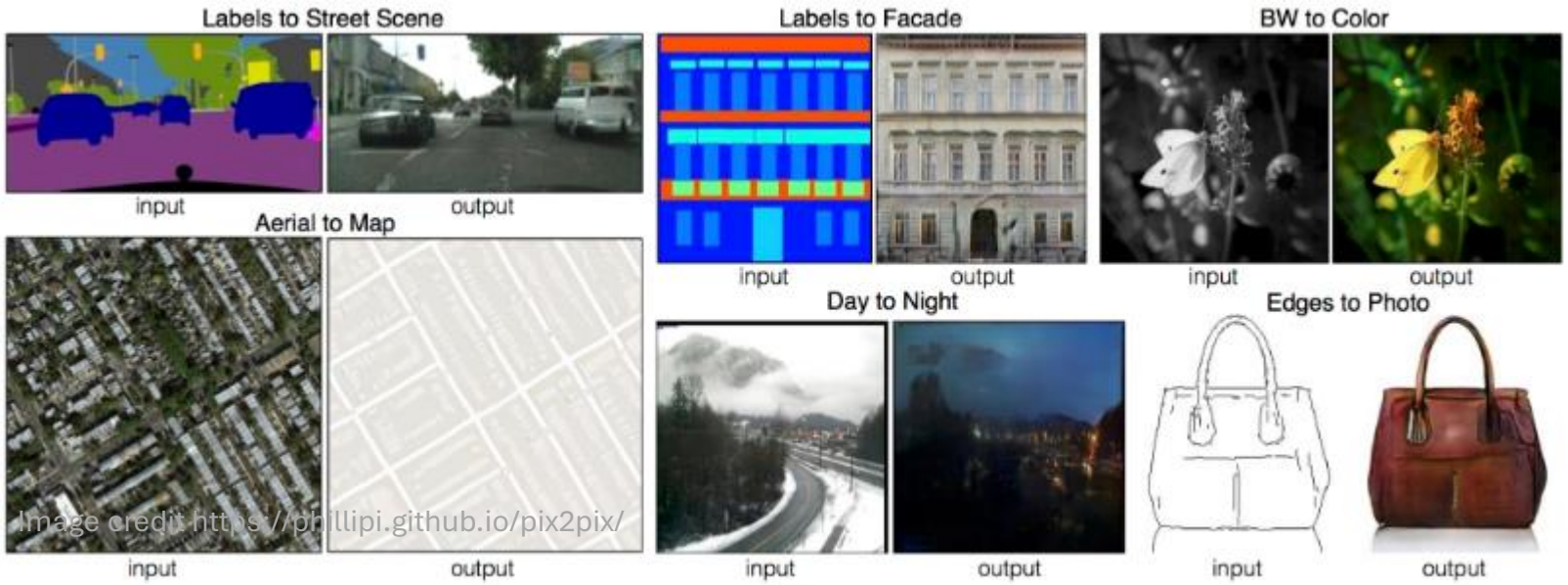


Image credit https://phillipi.github.io/pix2pix/

Isola et al. „Image to Image translation with conditional adversarial networks“, CVPR‘17

Generative Models

CycleGAN : Unpaired image-to-image translation

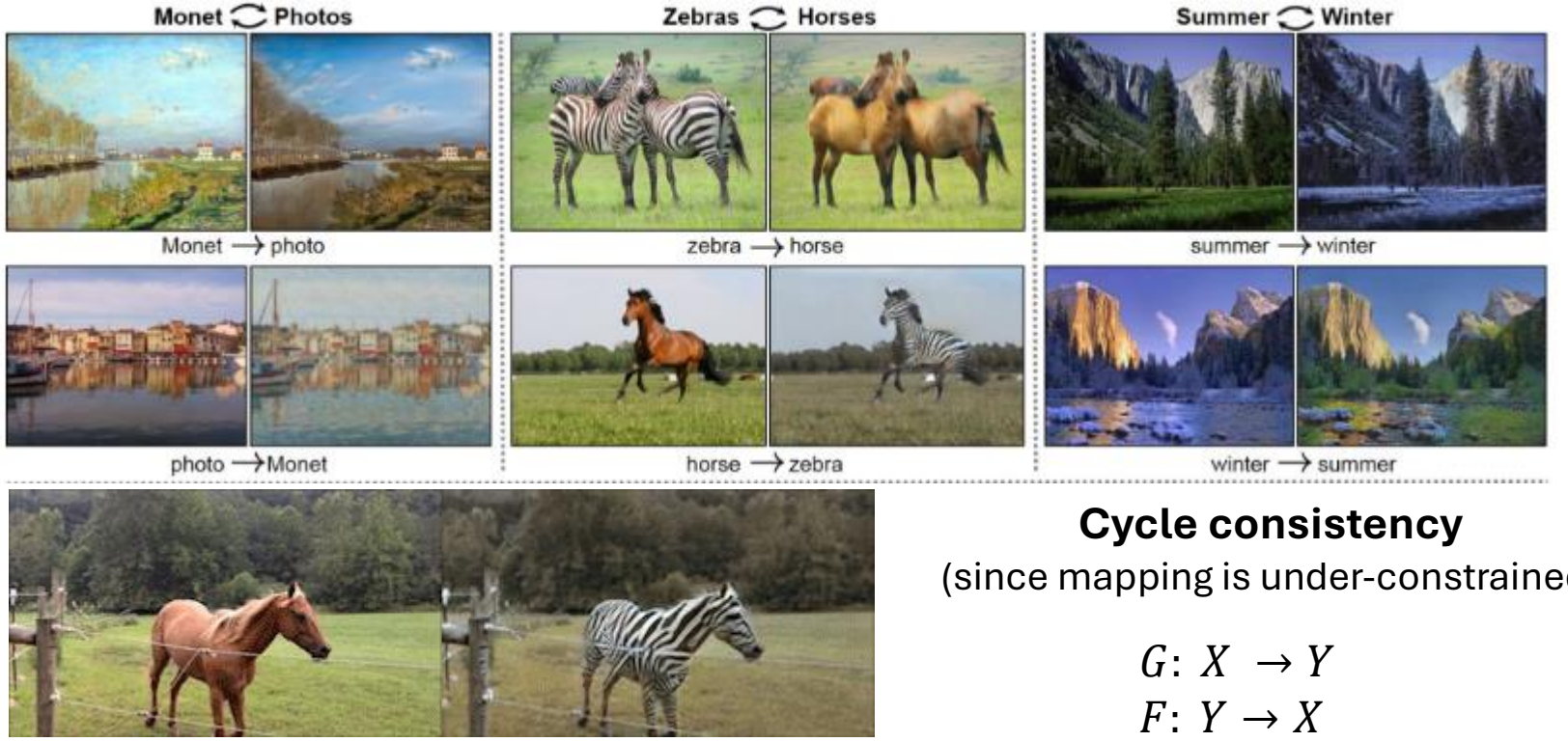


Image credit: https://github.com/junyanz/CycleGAN

Cycle consistency
(since mapping is under-constrained)

$$G\colon X \to Y$$
$$F\colon Y \to X$$
$$F\big(G(X)\big) \approx X$$

Zhu et al., „Unpaired image-to-image translation using cycle-consistent adversarial networks“, ICCV‘17

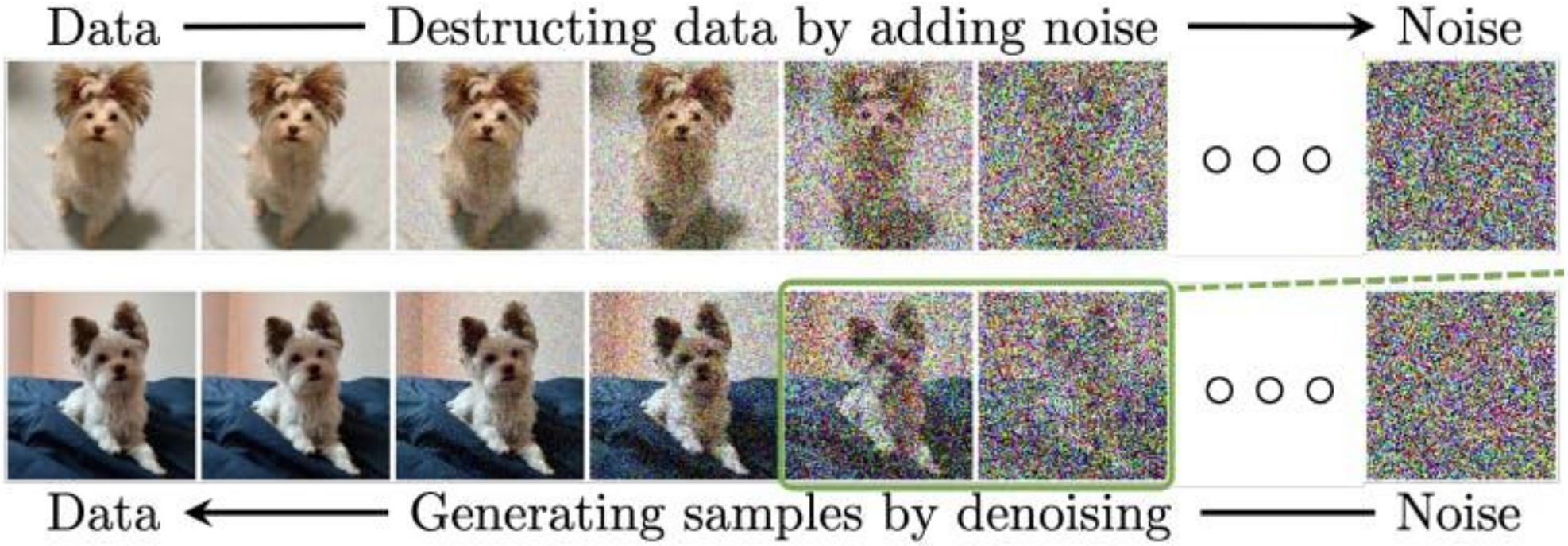
Generative Models

Image completion / Inpainting

Iizuka et al., „Globally and Locally Consistent Image Completion", SIGGRAPH'17

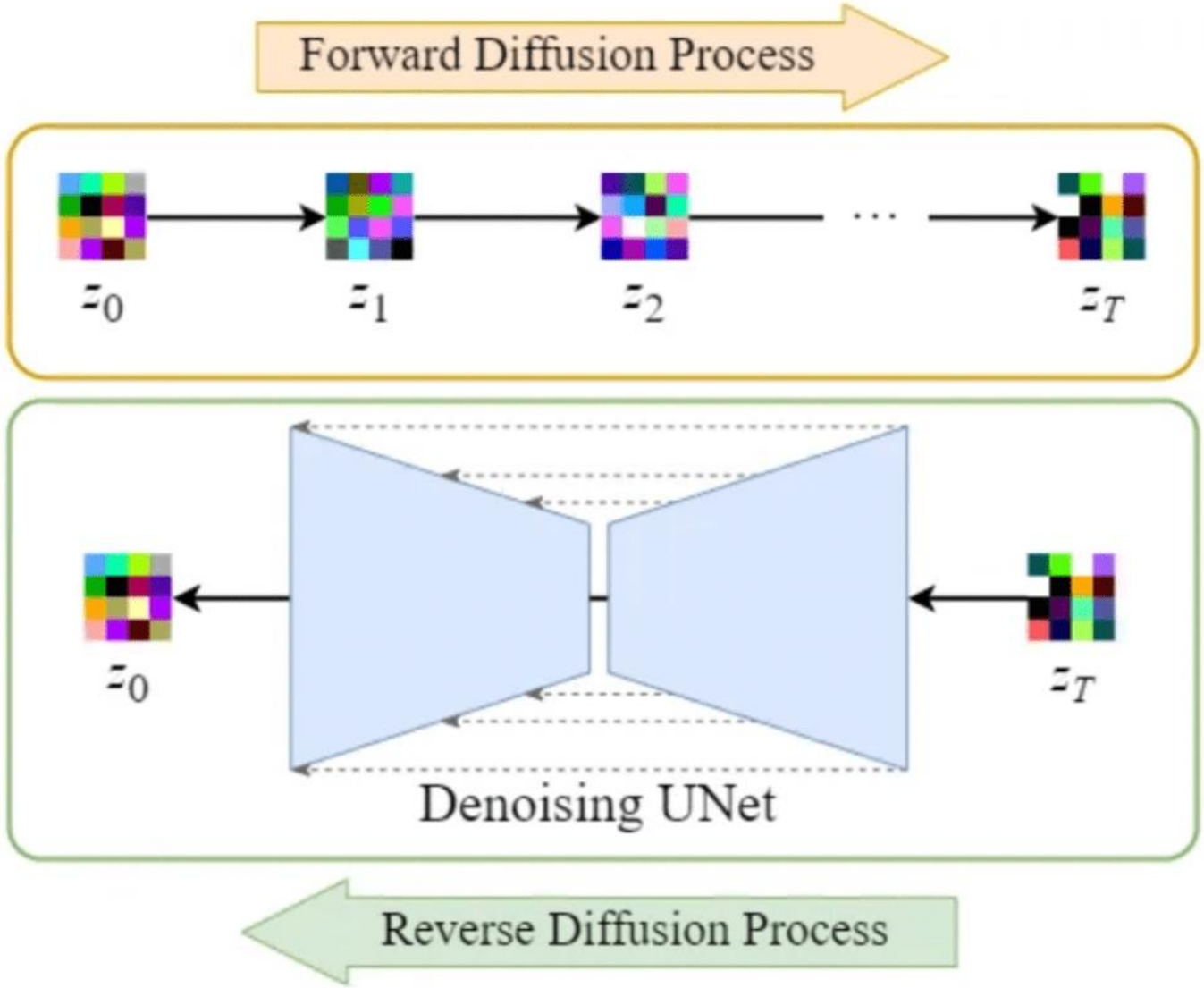
Generative models

Diffusion models



Generative models

Diffusion models

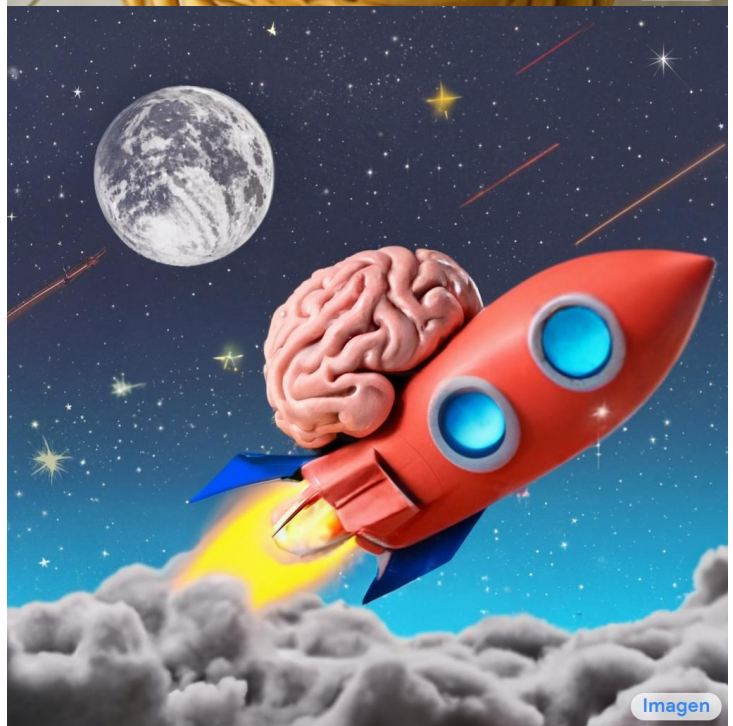


Try results:
https://stablediffusionweb.com/

Generative models

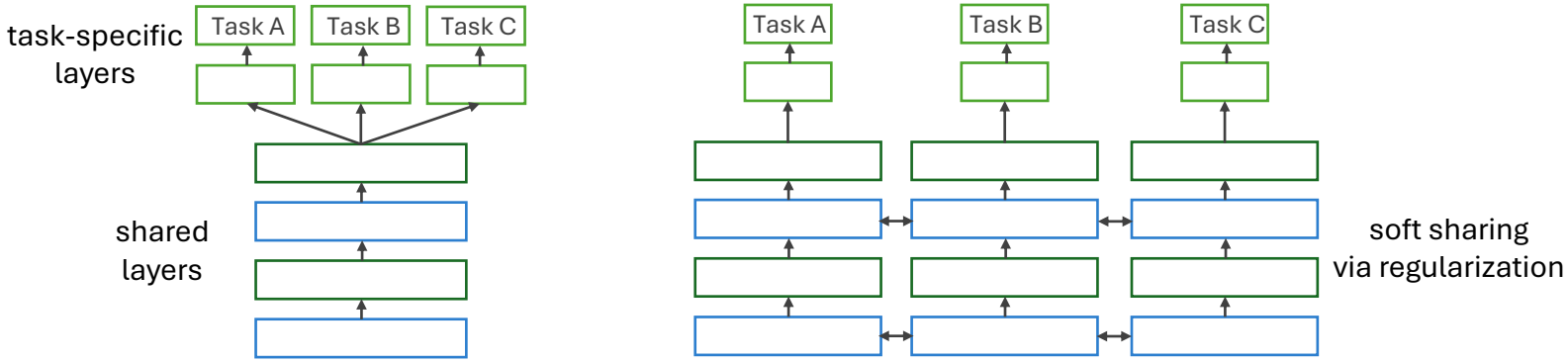
Diffusion models

https://imagen.research.google/



Multi-task Learning

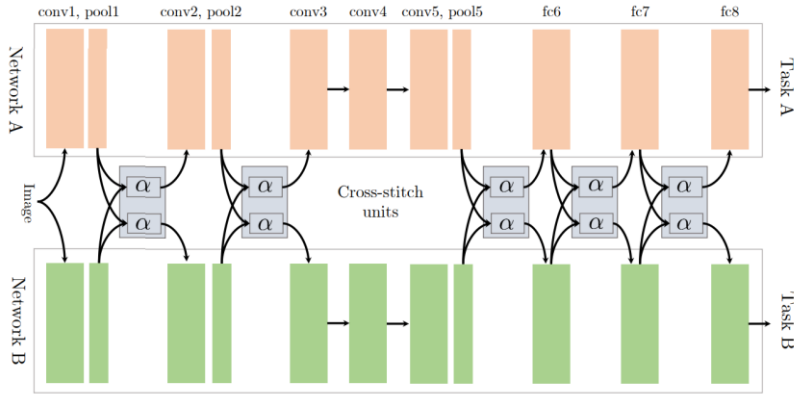
- Jointly learn different tasks with the same architecture
- Differs from transfer learning (which is sequential)



Cross-stitch networks

Sharing unit:

$$\begin{bmatrix} \tilde{x}_{\mathrm{A}}^{ij} \\ \tilde{x}_{\mathrm{B}}^{ij} \end{bmatrix} = \begin{bmatrix} \alpha_{\mathrm{AA}} & \alpha_{\mathrm{AB}} \\ \alpha_{BA} & \alpha_{\mathrm{BB}} \end{bmatrix} \begin{bmatrix} x_{\mathrm{A}}^{ij} \\ x_{\mathrm{B}}^{ij} \end{bmatrix}$$



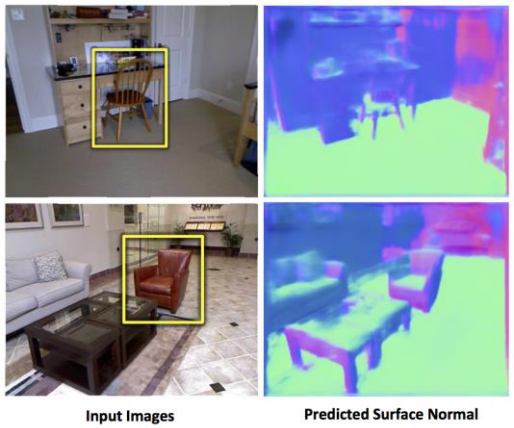
Hard sharing:
Choose where to split

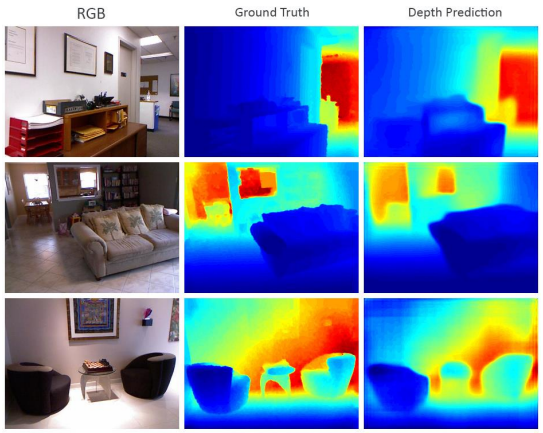
vs

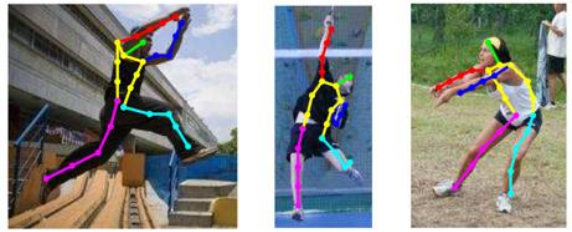
Soft sharing:
Learn what to share

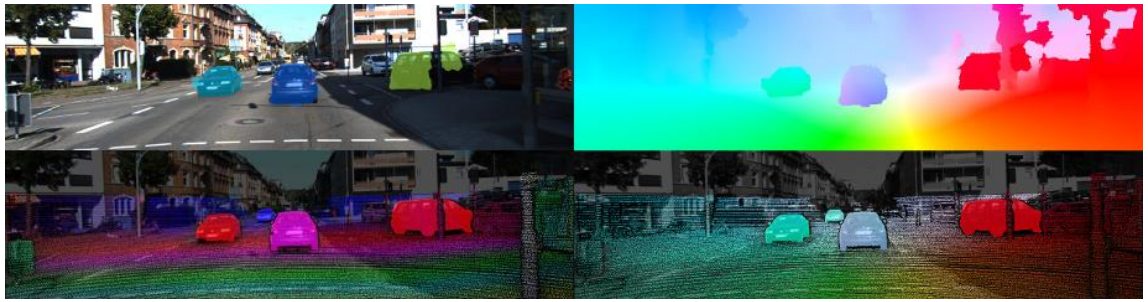
Misra et al. “Cross-stitch Networks for Multi-task Learning”, CVPR’16

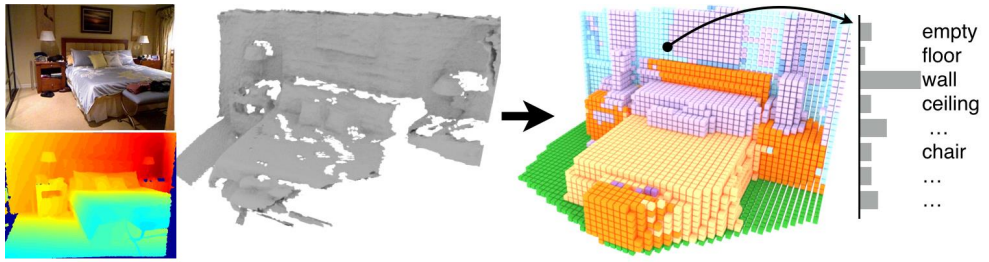
Many more Computer Vision Tasks...











Recap

- Convolution Networks are suitable for processing images
- The core building blocks are convolutions, pooling, activation and normalization layers
- We discussed some commonly used architectures
- They enable endless computer vision tasks
  - Image classification, object detection, segmentation, relationship prediction
  - Image generation and editing
- Today:
  - More towards inferring 3D modalities from images
  - Deep Learning for 3D data

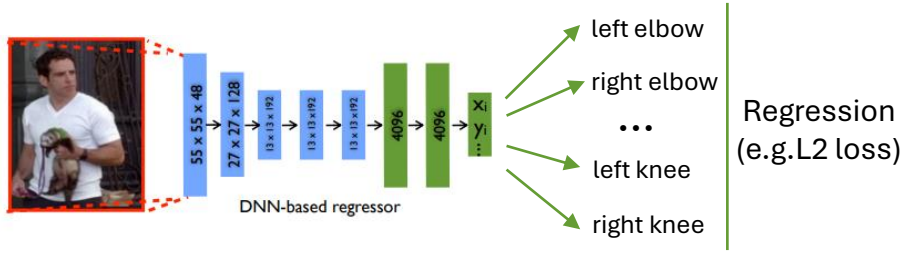
Towards 3D understanding of images

Deep Learning for 3D

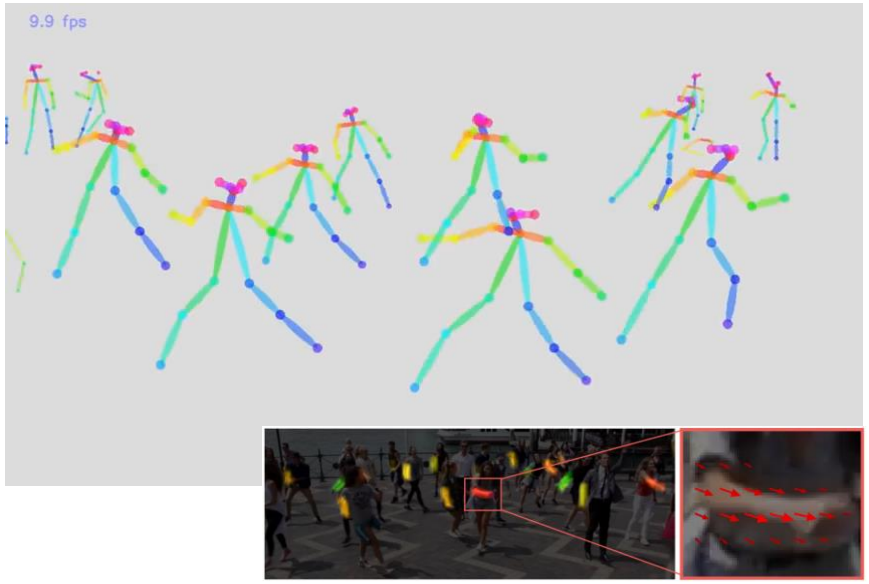
Lecture 3

Towards 3D understanding of images

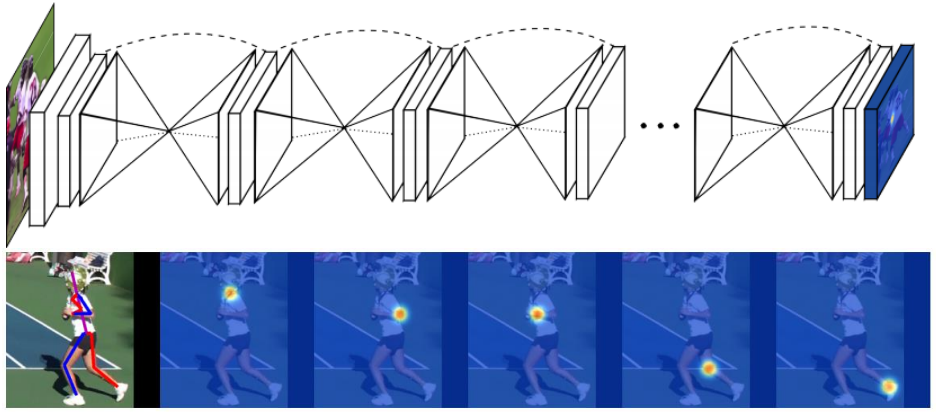
Human Pose Estimation



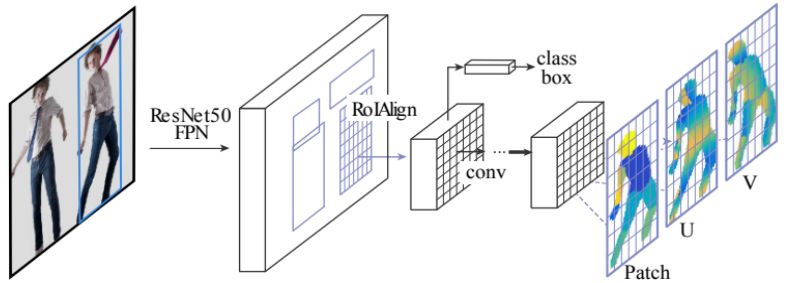
Toshev et al. “DeepPose: Human Pose Estimation via Deep Neural Networks”, CVPR’14



Cao et al. “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”, CVPR’17



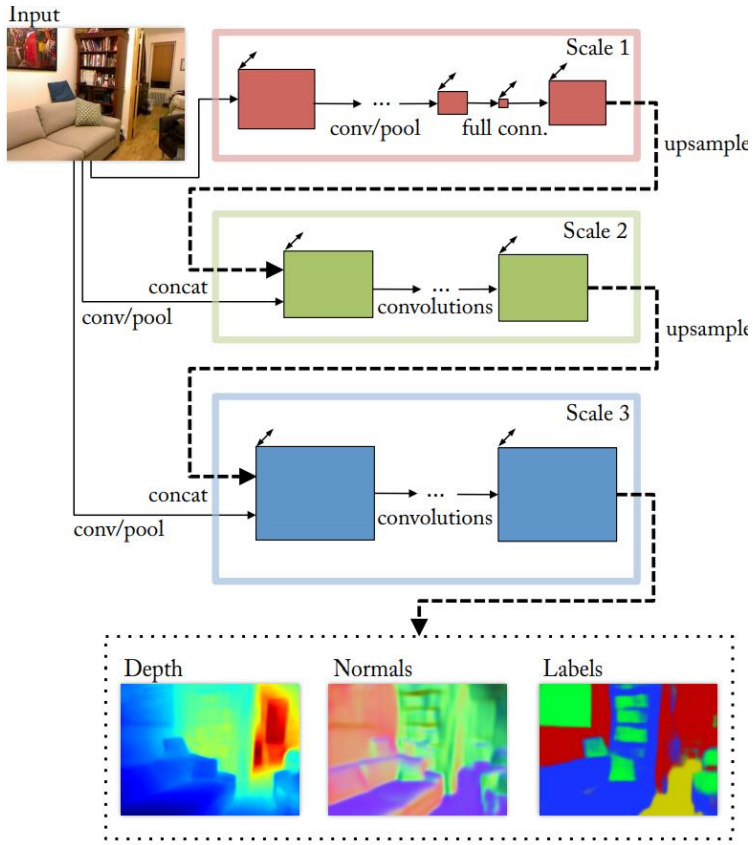
Newell et al. “Stacked Hourglass Networks for Human Pose Estimation”, ECCV’16



Güler et al. “DensePose: Dense Human Pose Estimation in The Wild”, CVPR’18

Depth Estimation

Supervised (from RGB-D data)



Coarse-to-fine prediction with multiple scales

- Fully connected architecture (VGG) as a first coarse scale
- Reshape output to low resolution H×W map
- Add more convolutions for refinement
- Predict more tasks, e.g. surface normal and semantic labels

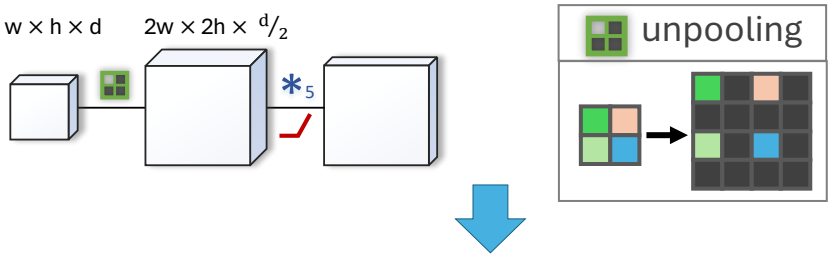
Eigen and Fergus, “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture”, ICCV’15

Depth Estimation

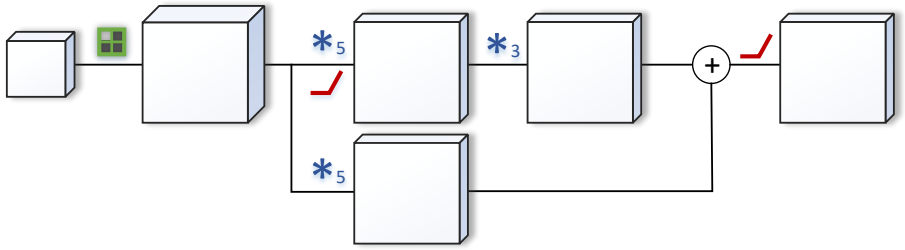
Supervised (from RGB-D data)

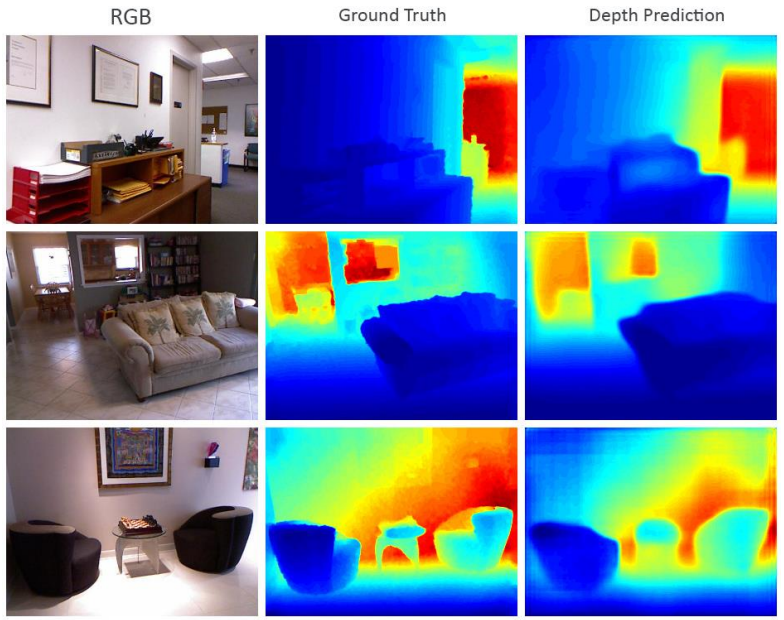
Fully convolutional ResNet

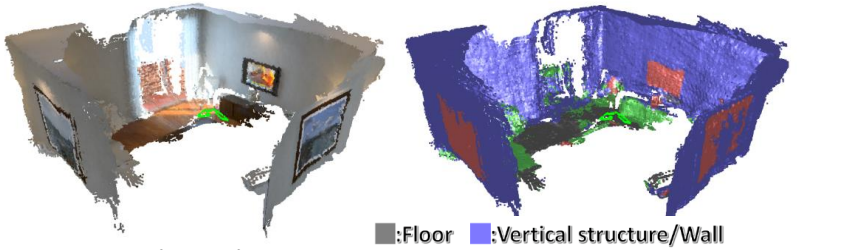
unpooling + convolution = “up-convolution”



up-convolution + residual leaning = “up-projection”





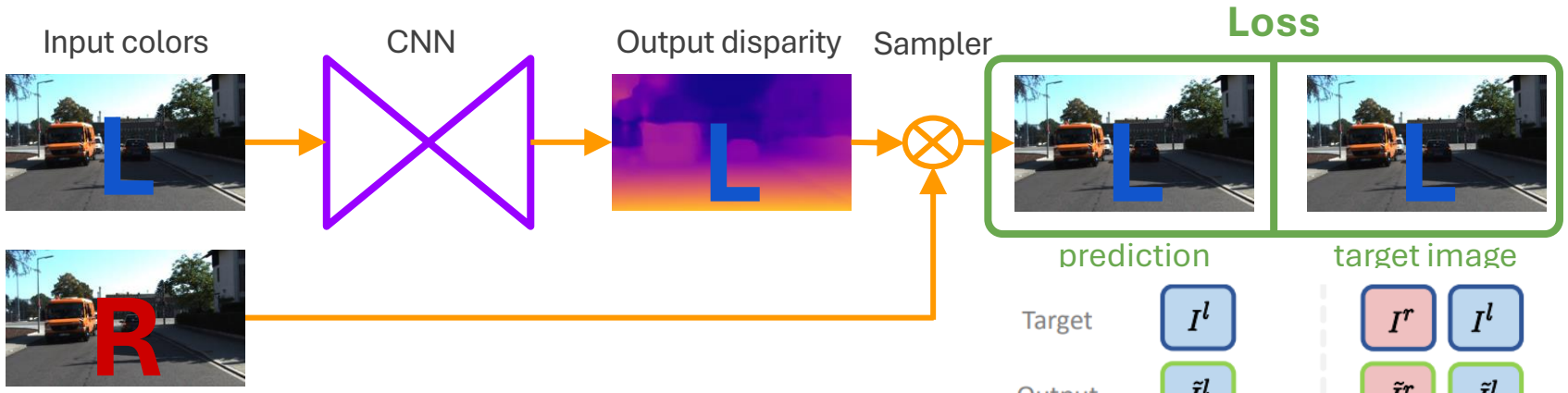


Application:
Monocular scene reconstruction

Laina et al., “Deeper Depth Prediction with Fully Convolutional Residual Networks”, 3DV’16

Depth Estimation

Unsupervised (from sequences and/or stereo data)

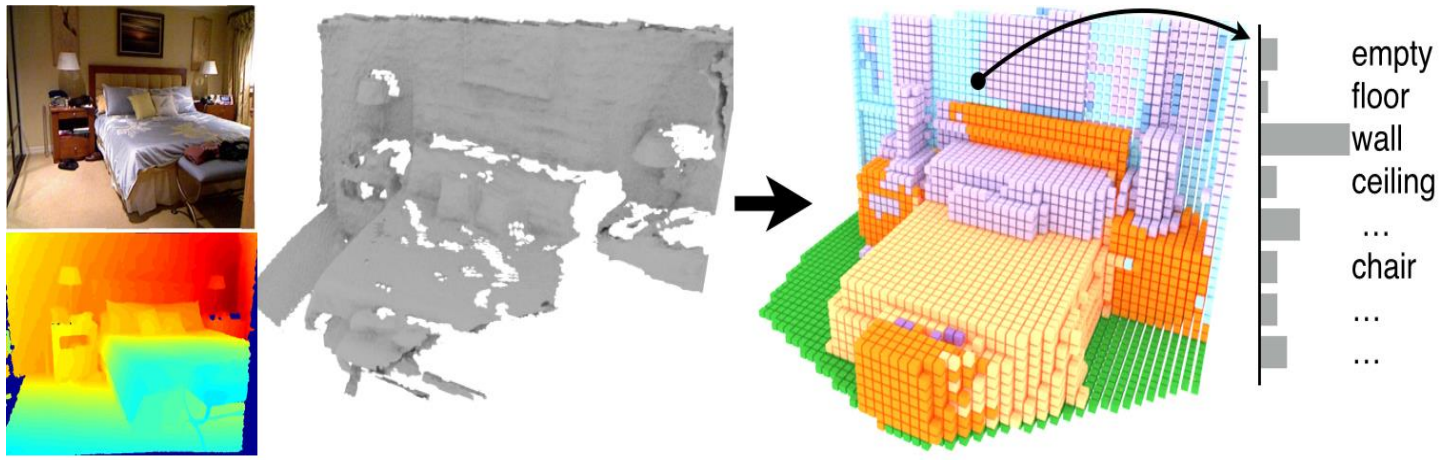


- Train with stereo pairs
- Discover disparity as intermediate representation and use it to warp the image

Add a left/right consistency loss

Godard et al., “Unsupervised Monocular Depth Estimation with Left-Right Consistency“, CVPR‘17

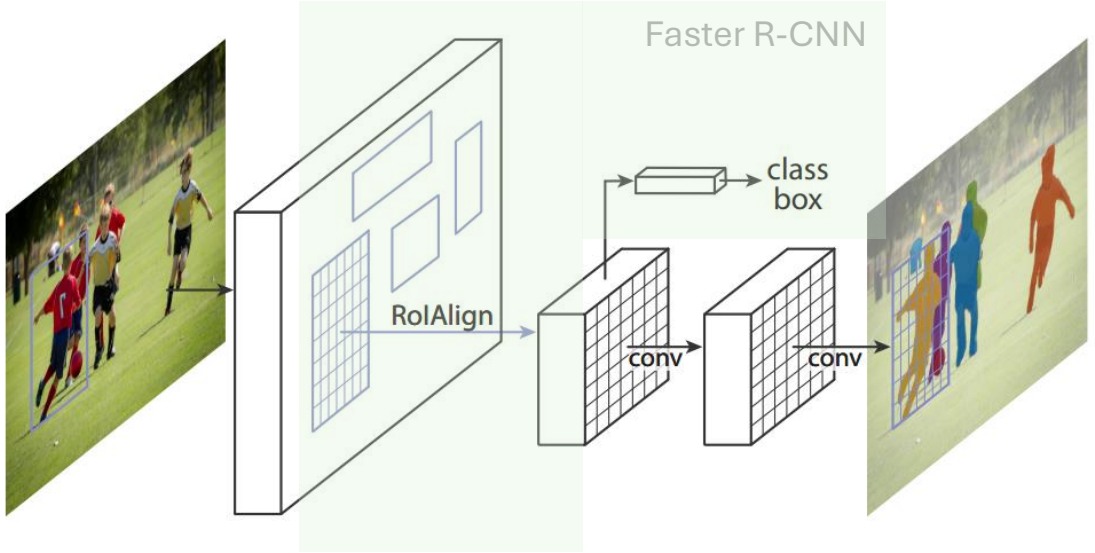
Semantic scene completion



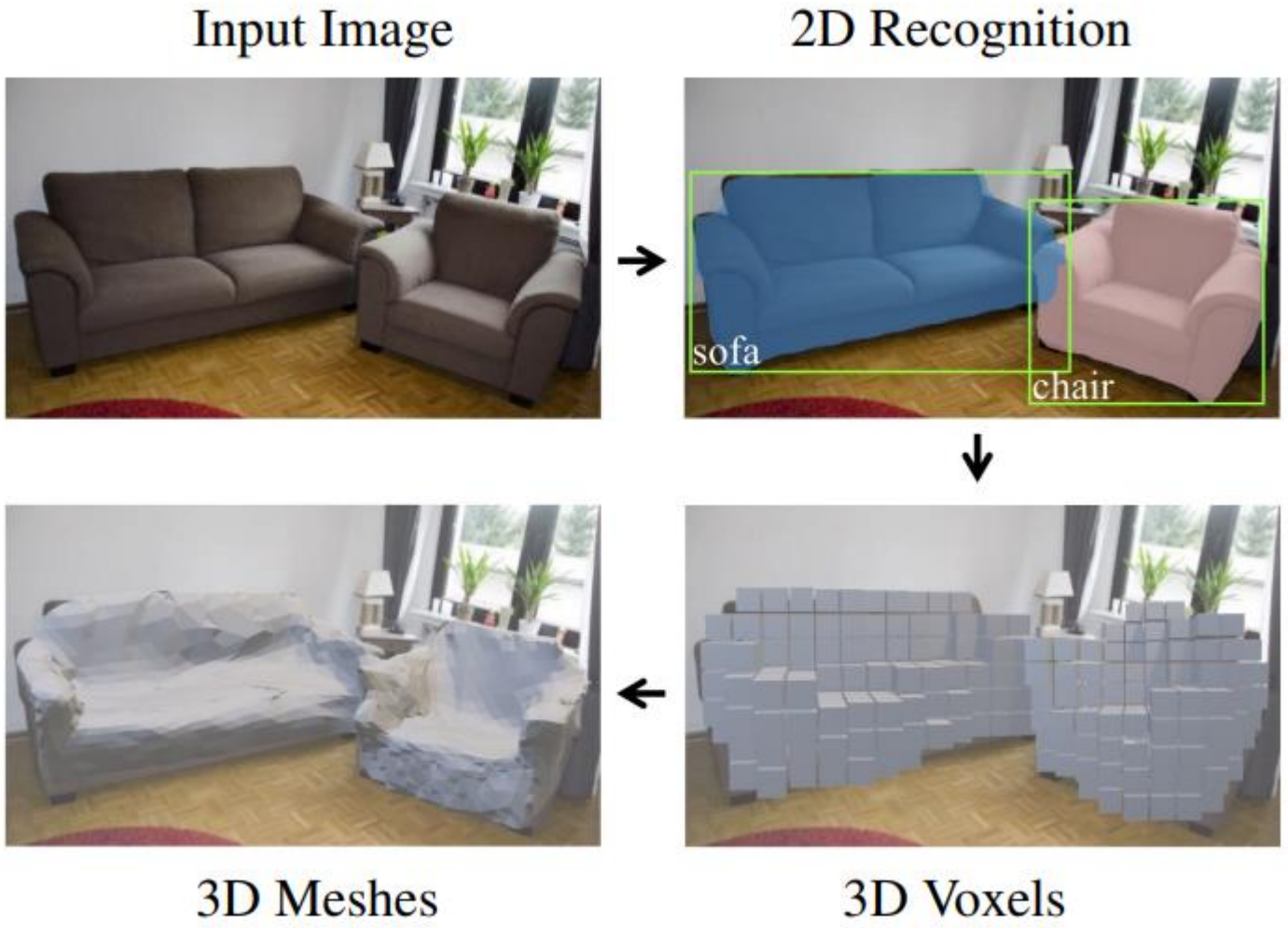
Song et al, “Semantic Scene Completion from a Single Depth Image”, CVPR’17

Mesh R-CNN

Remember Mask R-CNN?

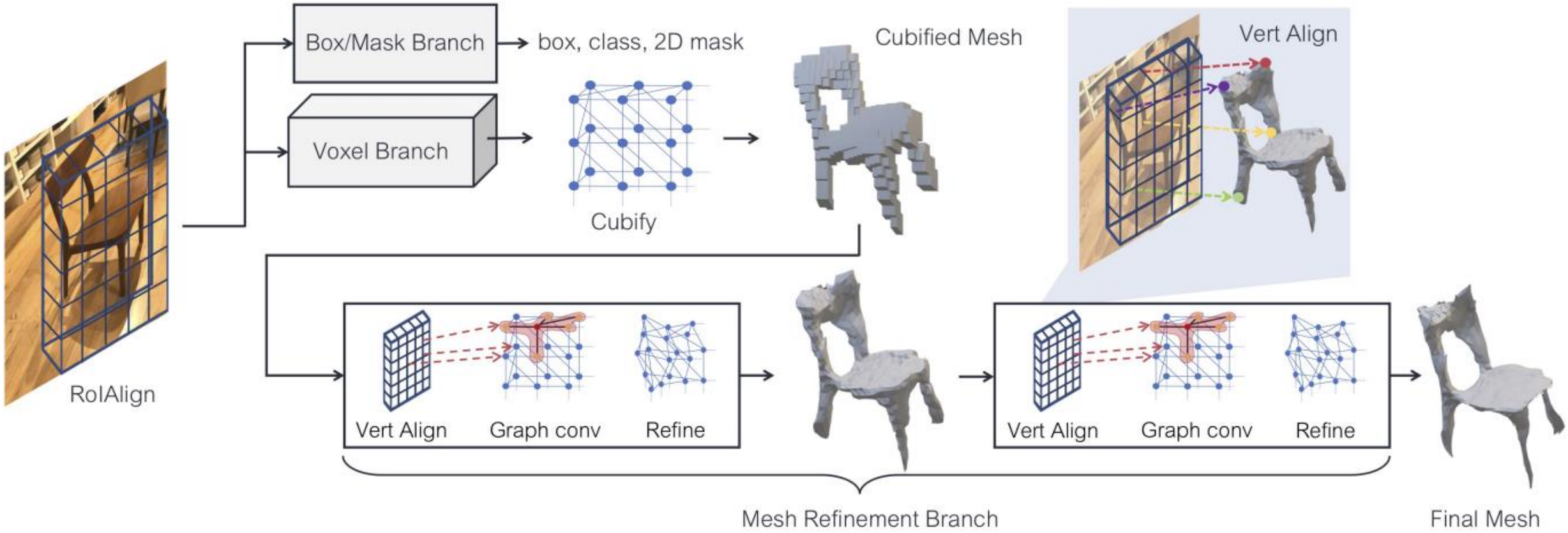


Mesh R-CNN



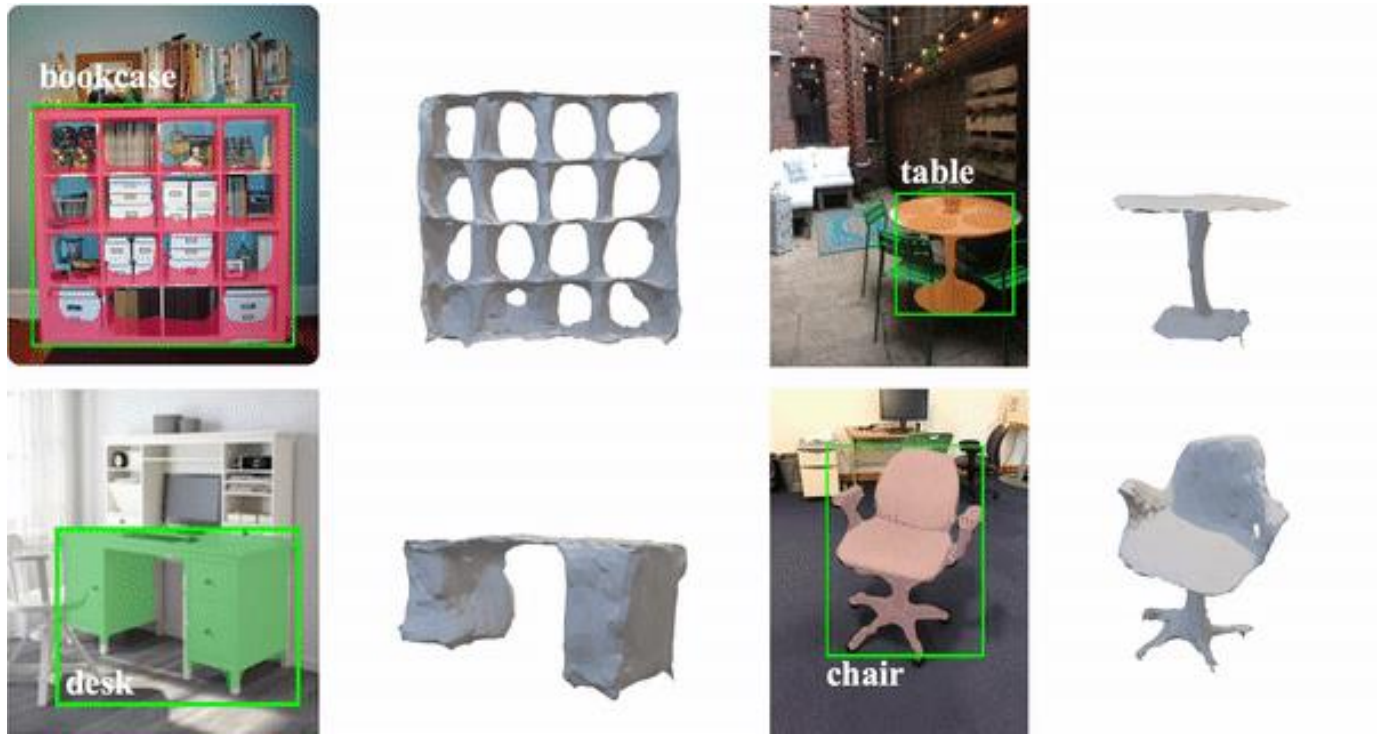
Mesh R-CNN Georgia Gkioxari Jitendra Malik Justin Johnson. ICCV 2019

Mesh R-CNN



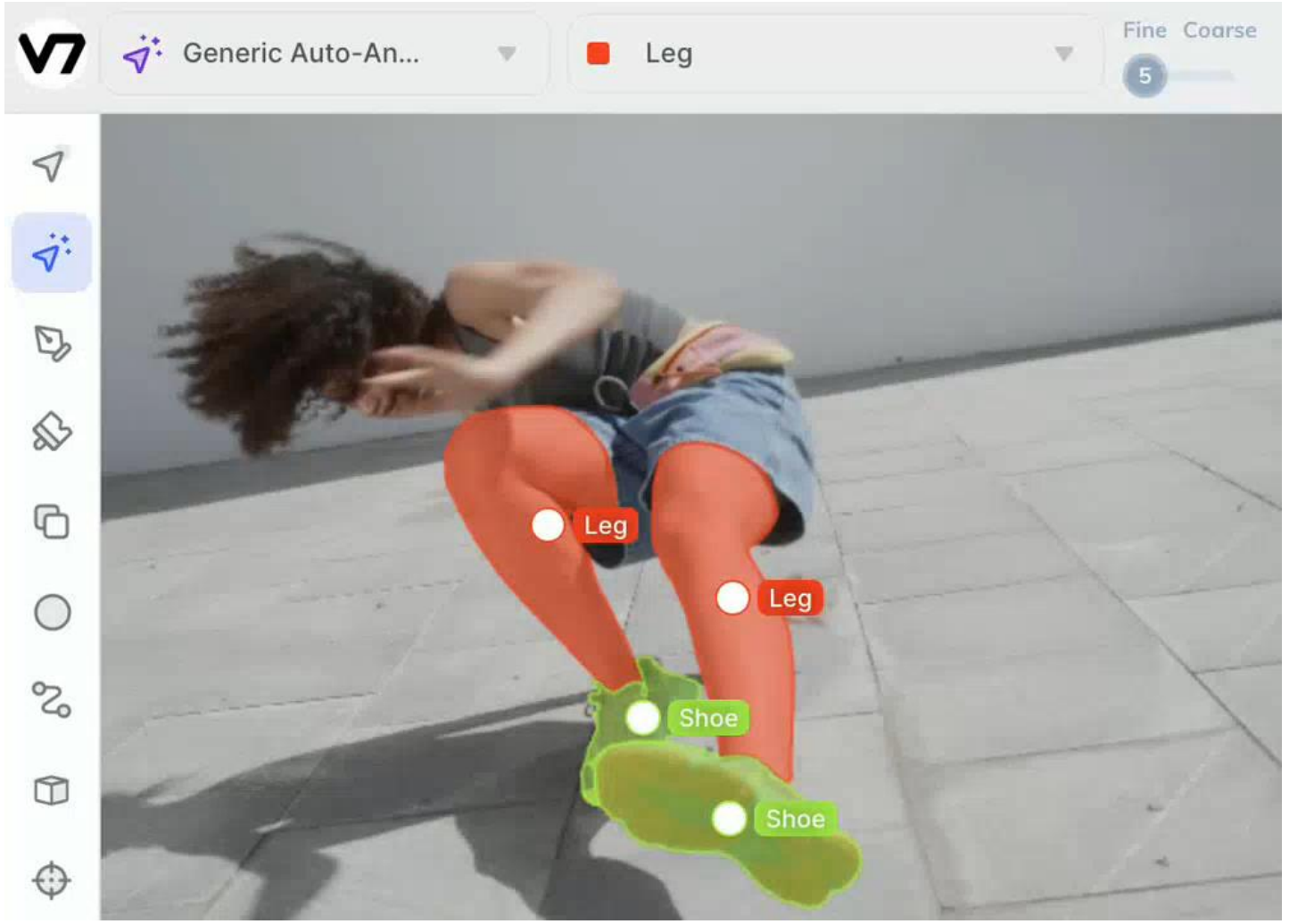
Mesh R-CNN Georgia Gkioxari Jitendra Malik Justin Johnson. ICCV 2019

Mesh R-CNN



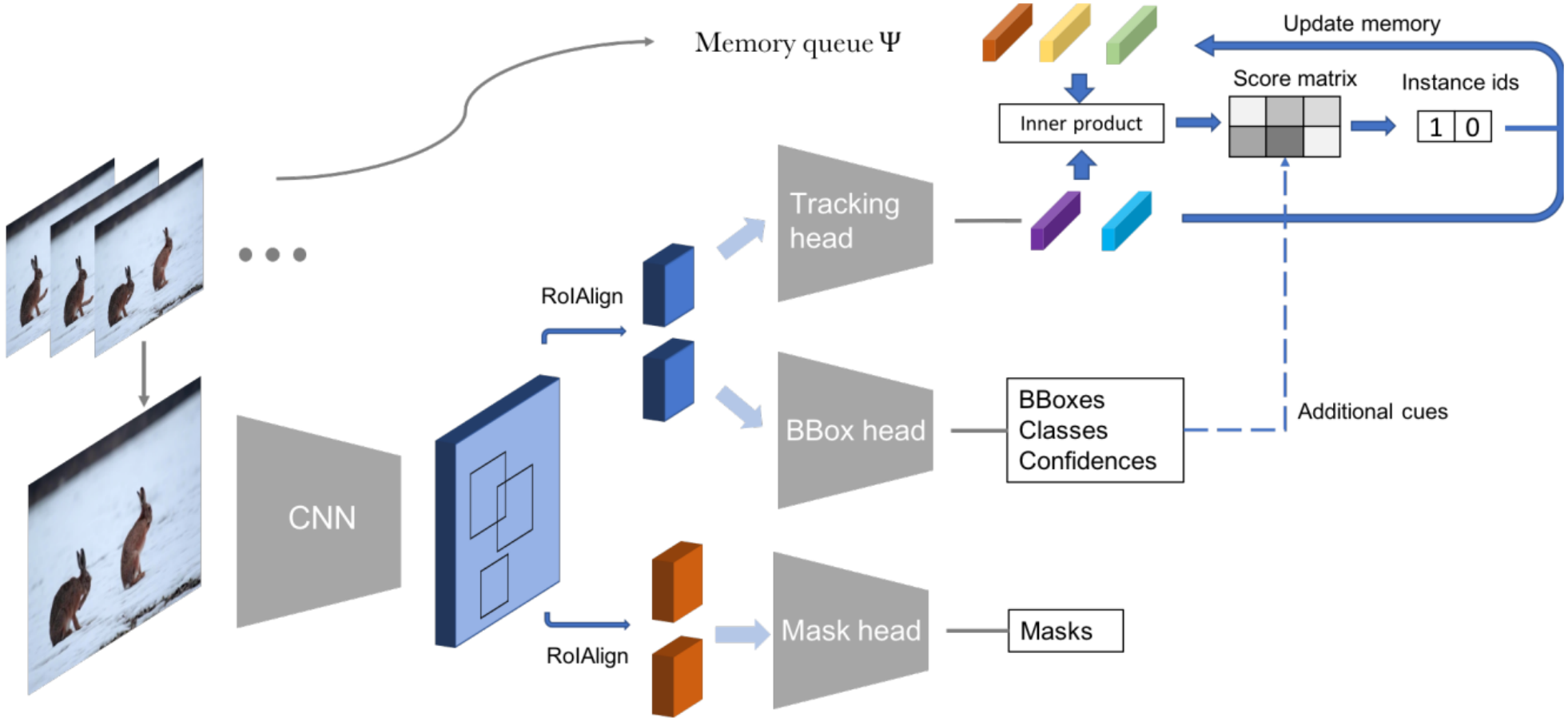
Mesh R-CNN Georgia Gkioxari Jitendra Malik Justin Johnson. ICCV 2019

Video Segmentation

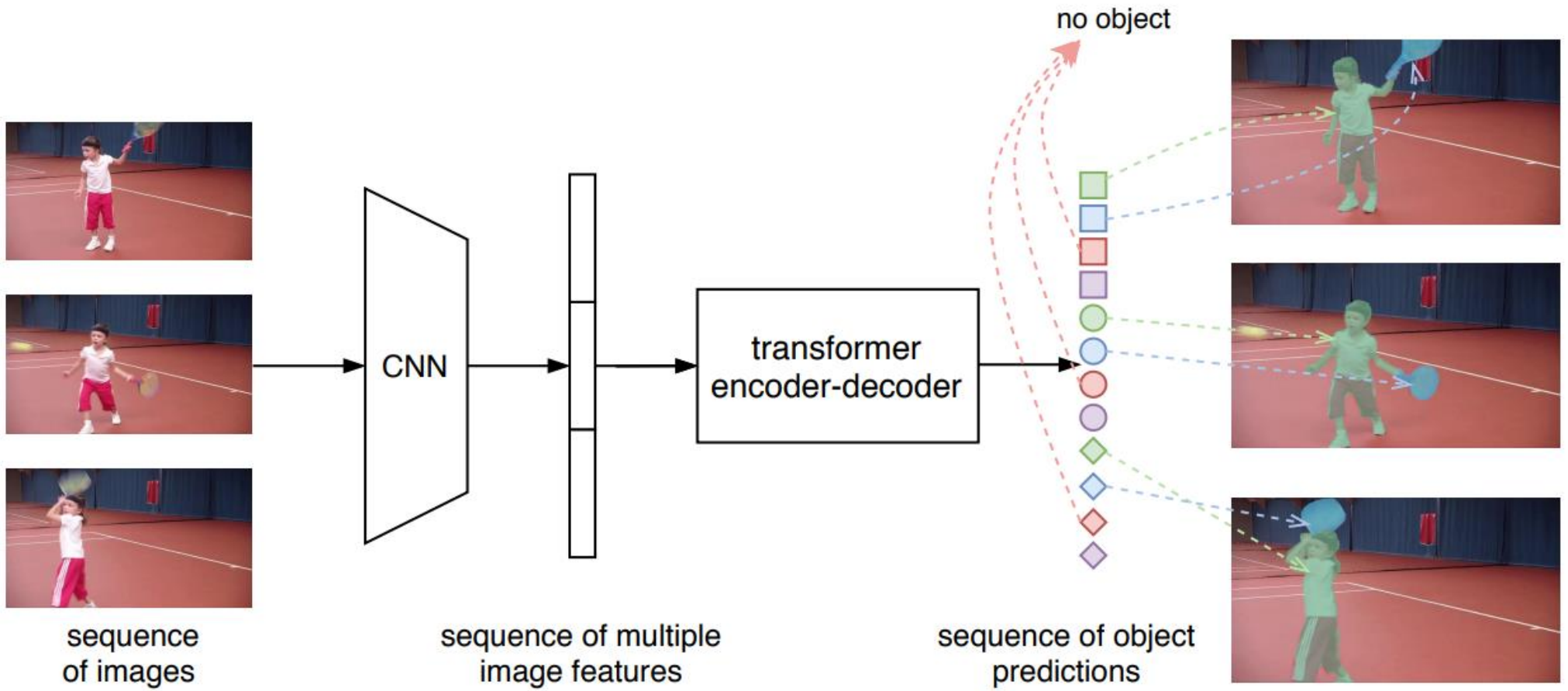


https://www.v7labs.com/blog/video-segmentation-guide

Video instance segmentation

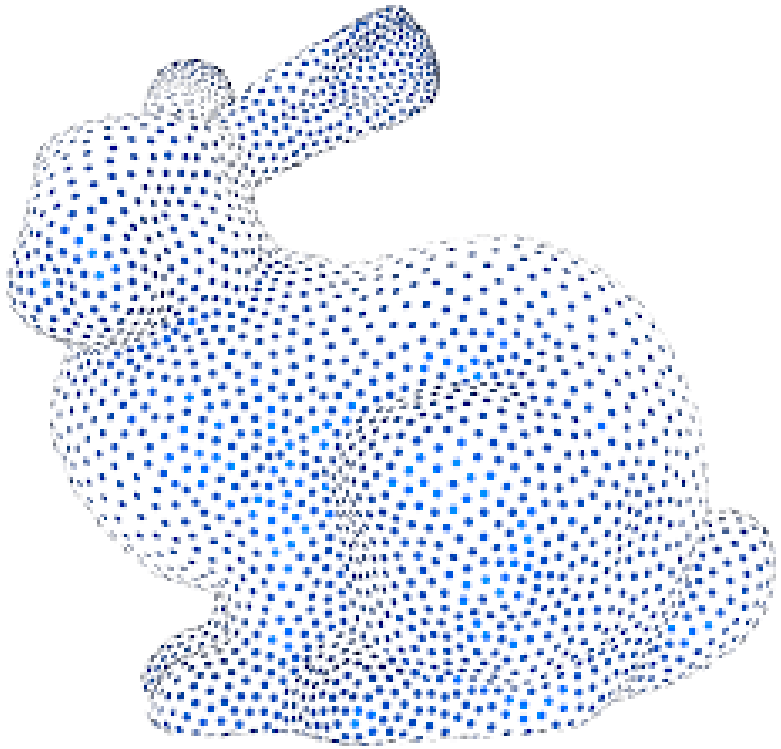


End-to-End Video Instance Segmentation with Transformers

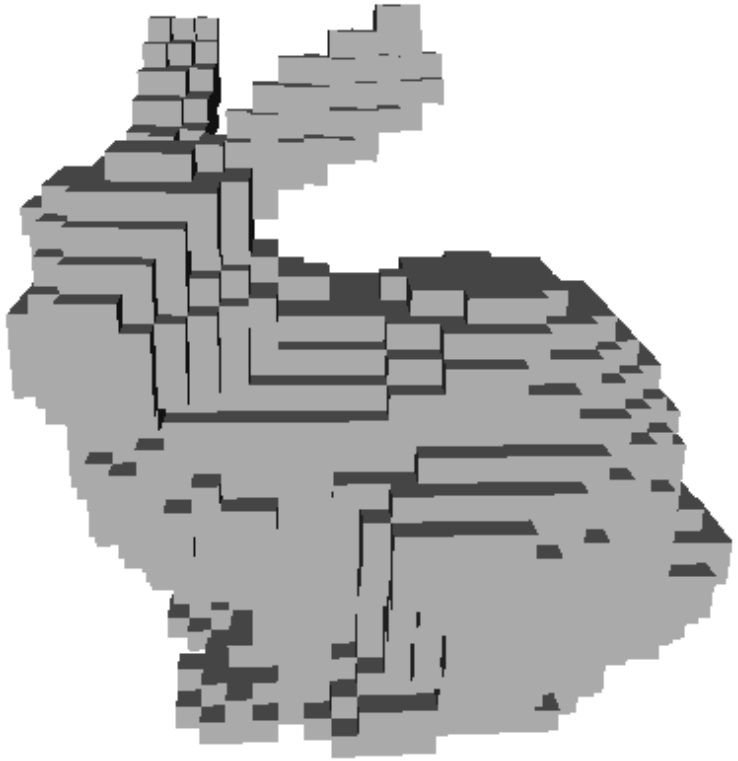


Deep Learning for 3D

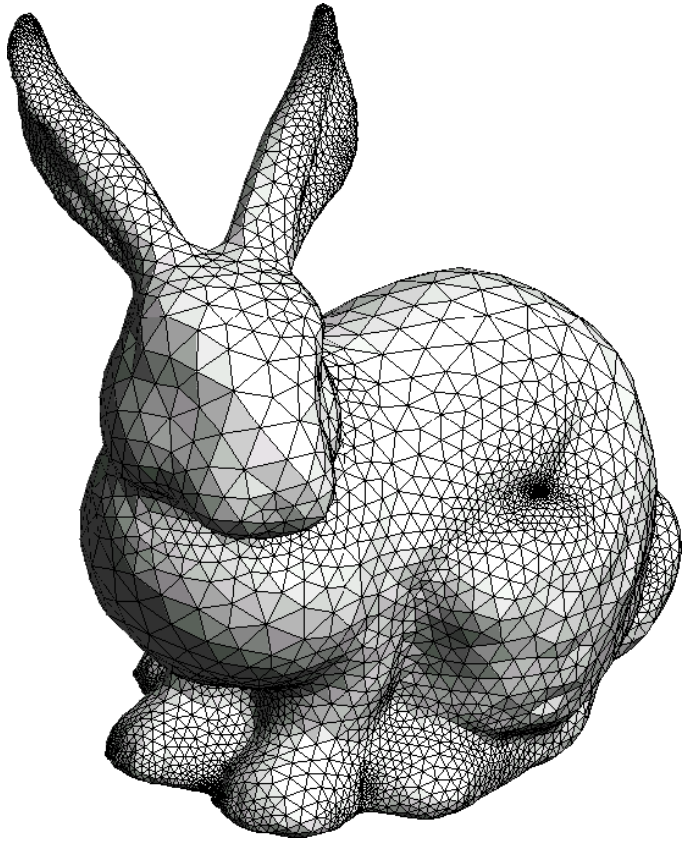
3D Data Representations



Point cloud



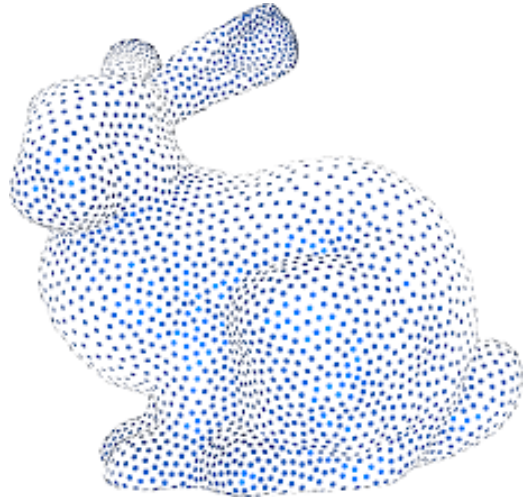
Voxel grids



mesh

Point Clouds

A set of 3D points (N x 3)



Pros

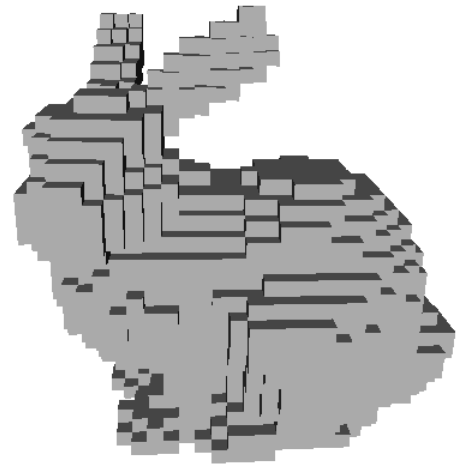
- **Efficient** coverage of space (only represent non-empty space)

Cons

- Connectivity is not given (need to run nearest neighbours to identify closest point)

Voxel grids

Binary occupancy map of size N x M x K
1s indicate presence of an object



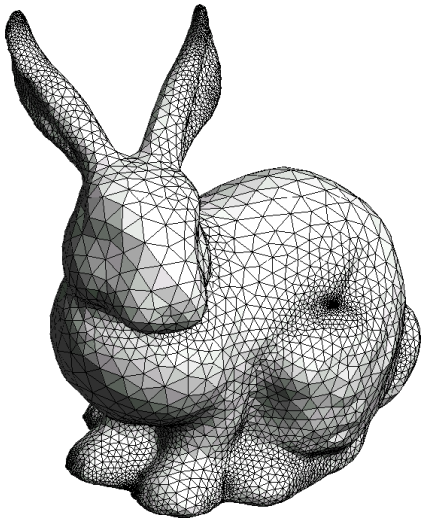
Pros

- Connectivity is known, therefore easy to process, e.g. with 3D ConNets

Cons

- Not efficient, as one need to cover the whole 3D space, including large empty areas
- The object surface loses smoothness properties, due to resolution contraints

Meshes



A set of 3D points (N x 3) called **vertices**,
and a set of **faces** (usually triangles, M x 3) that connect the vertices

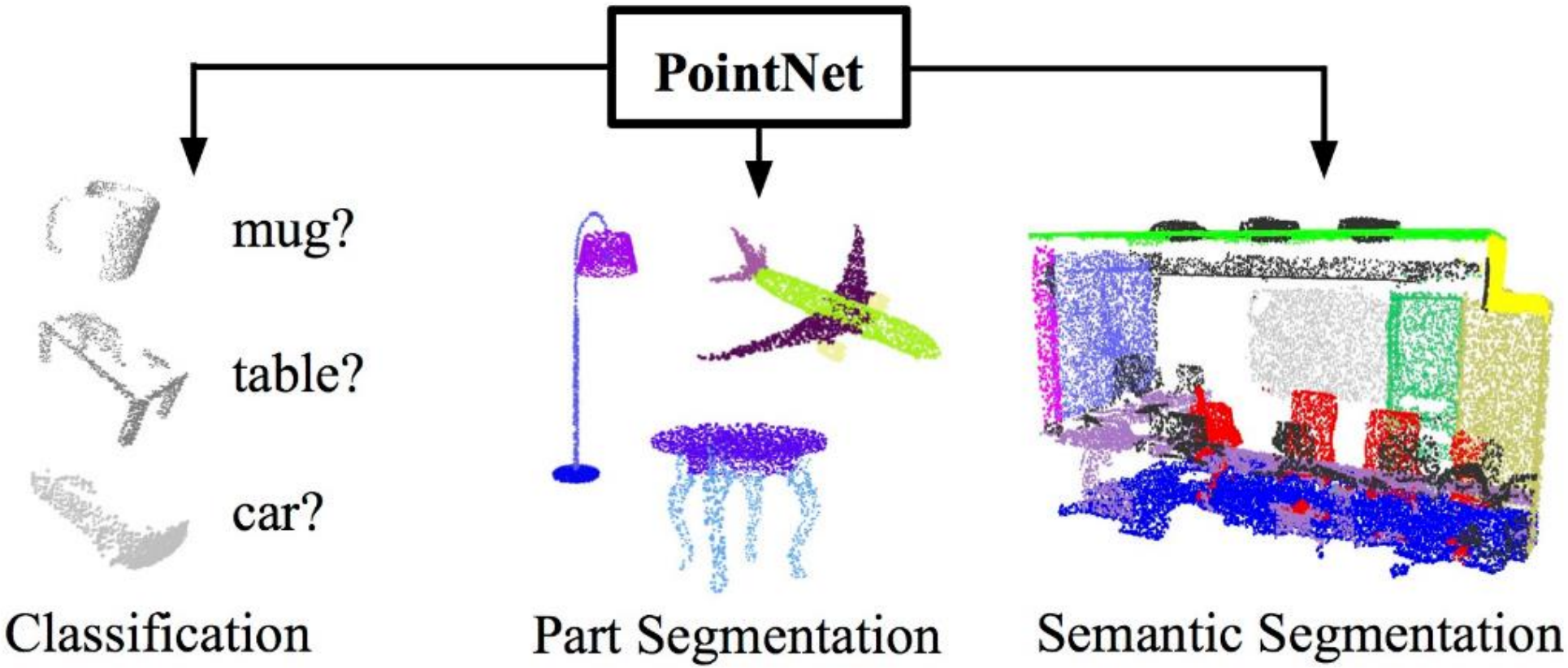
Pros

- Connectivity given via the faces
- More efficient than voxel grids
- Suitable for obtaining surface normals
- Can be textured

Cons

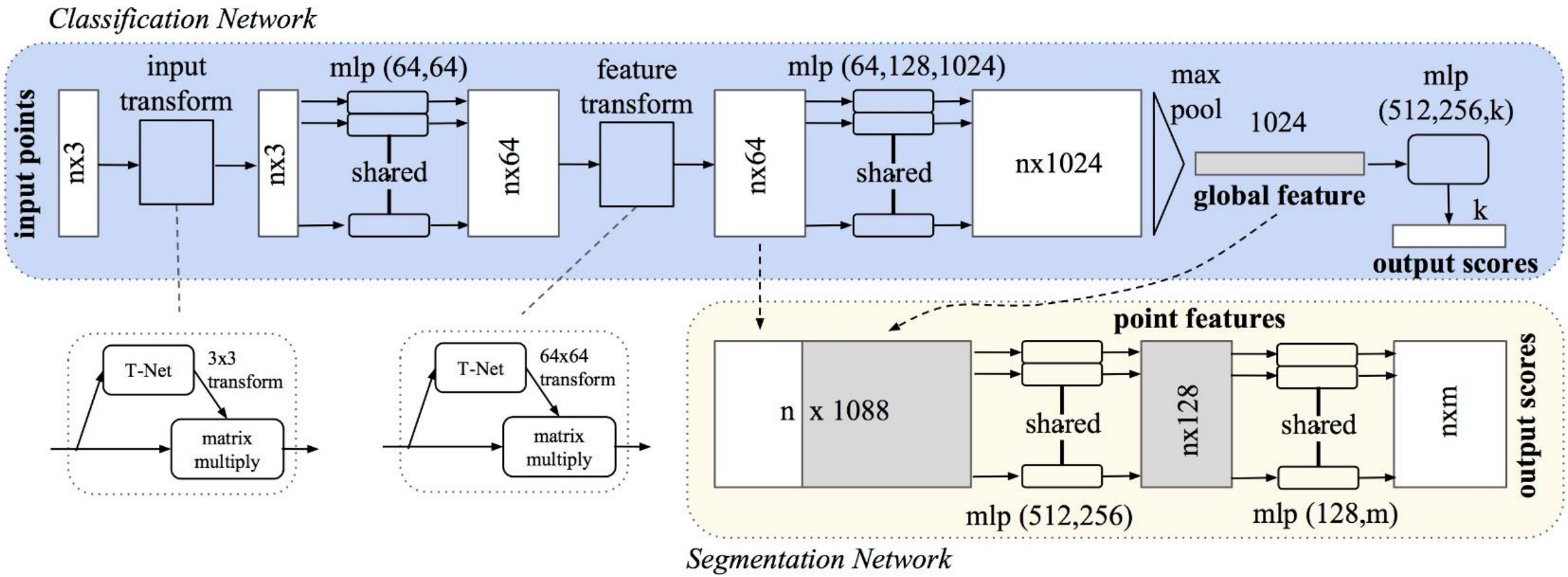
- Optimization not very easy
- Memory overhead compared to point clouds

Deep Learning for Point Clouds



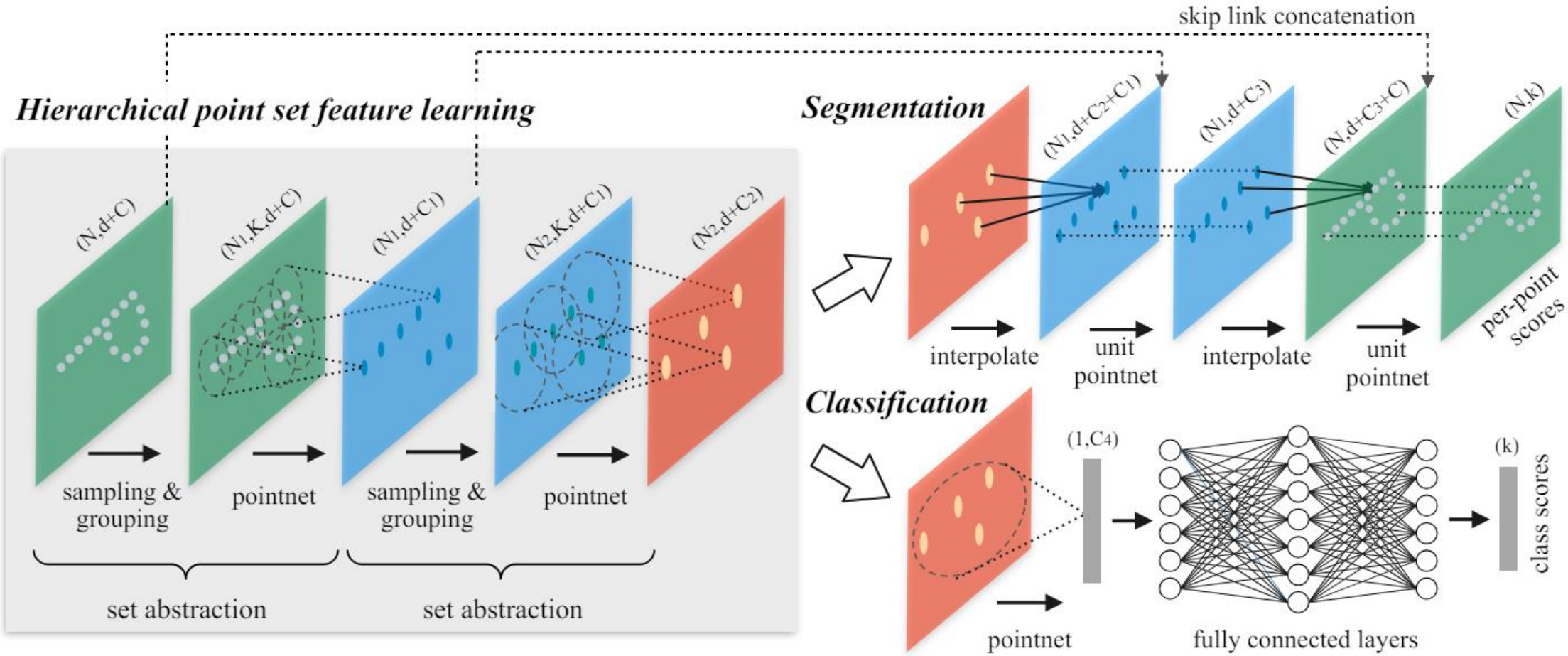
Deep Learning for Point Clouds

PointNet

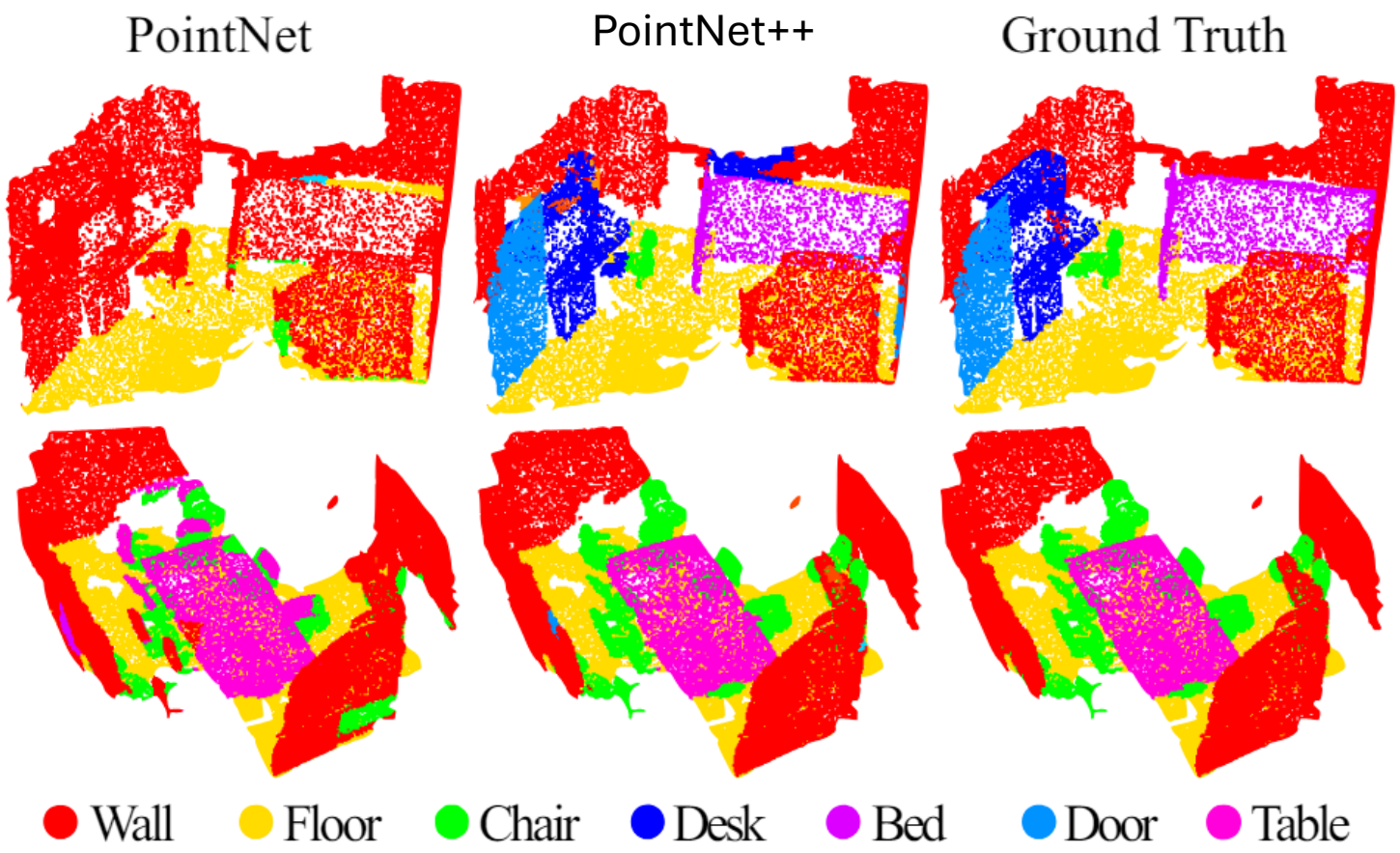


Deep Learning for Point Clouds

PointNet++

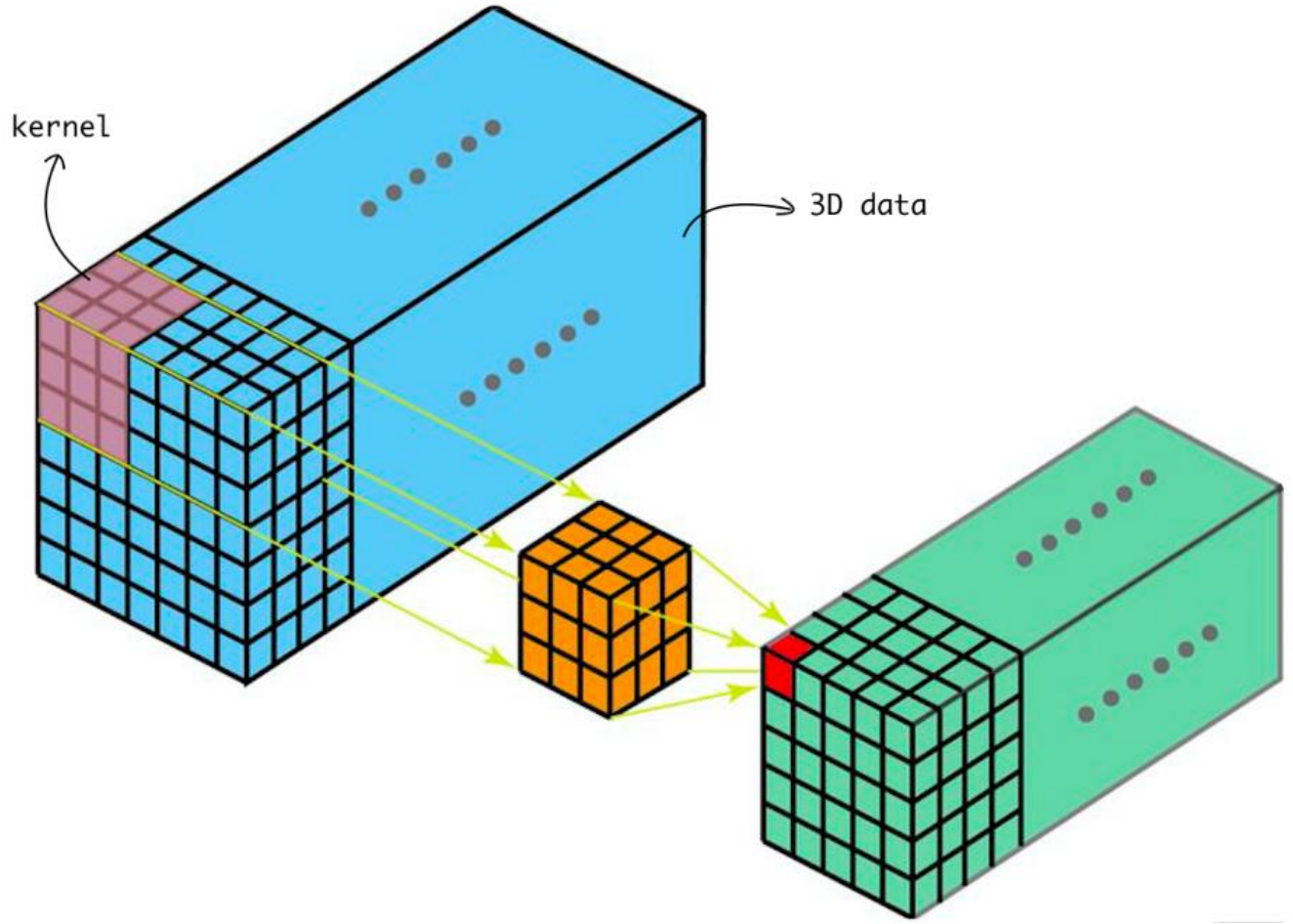


Deep Learning for Point Clouds



Deep Learning for Voxel Grids

3D Convolution

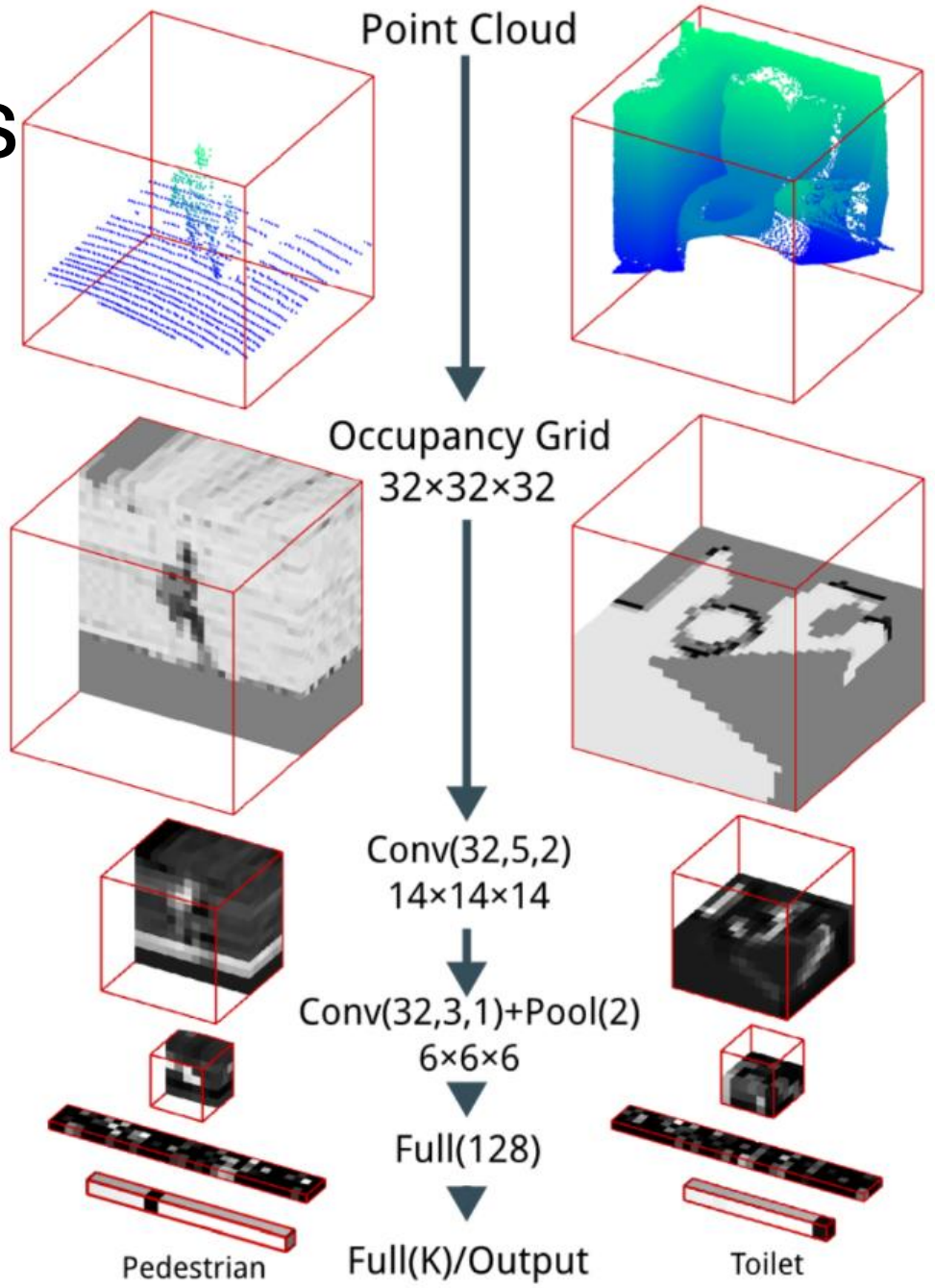


Deep Learning for Voxel Grids

VoxNet

Doesn’t scale well!

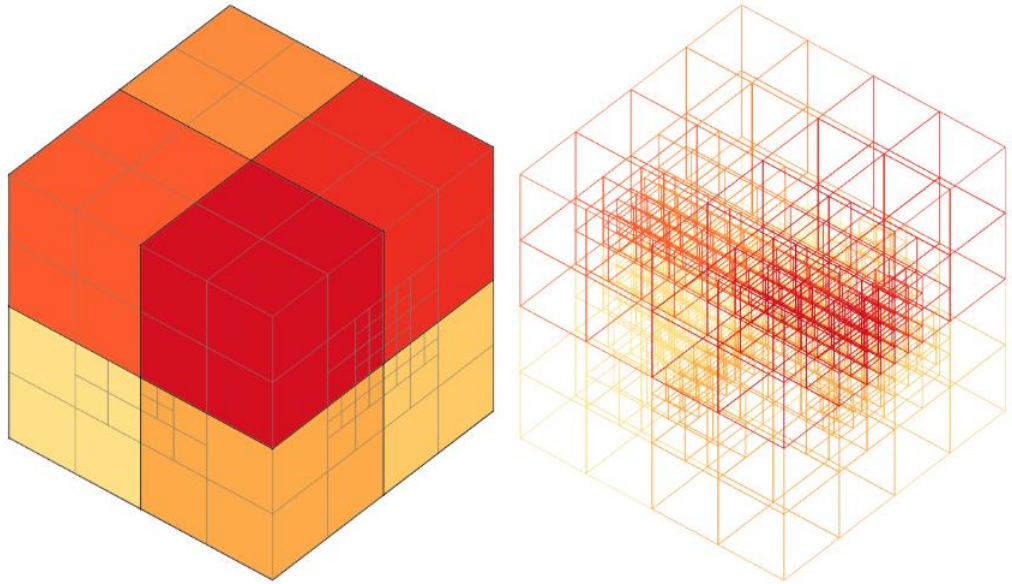
This limits the depth of the network that we can afford

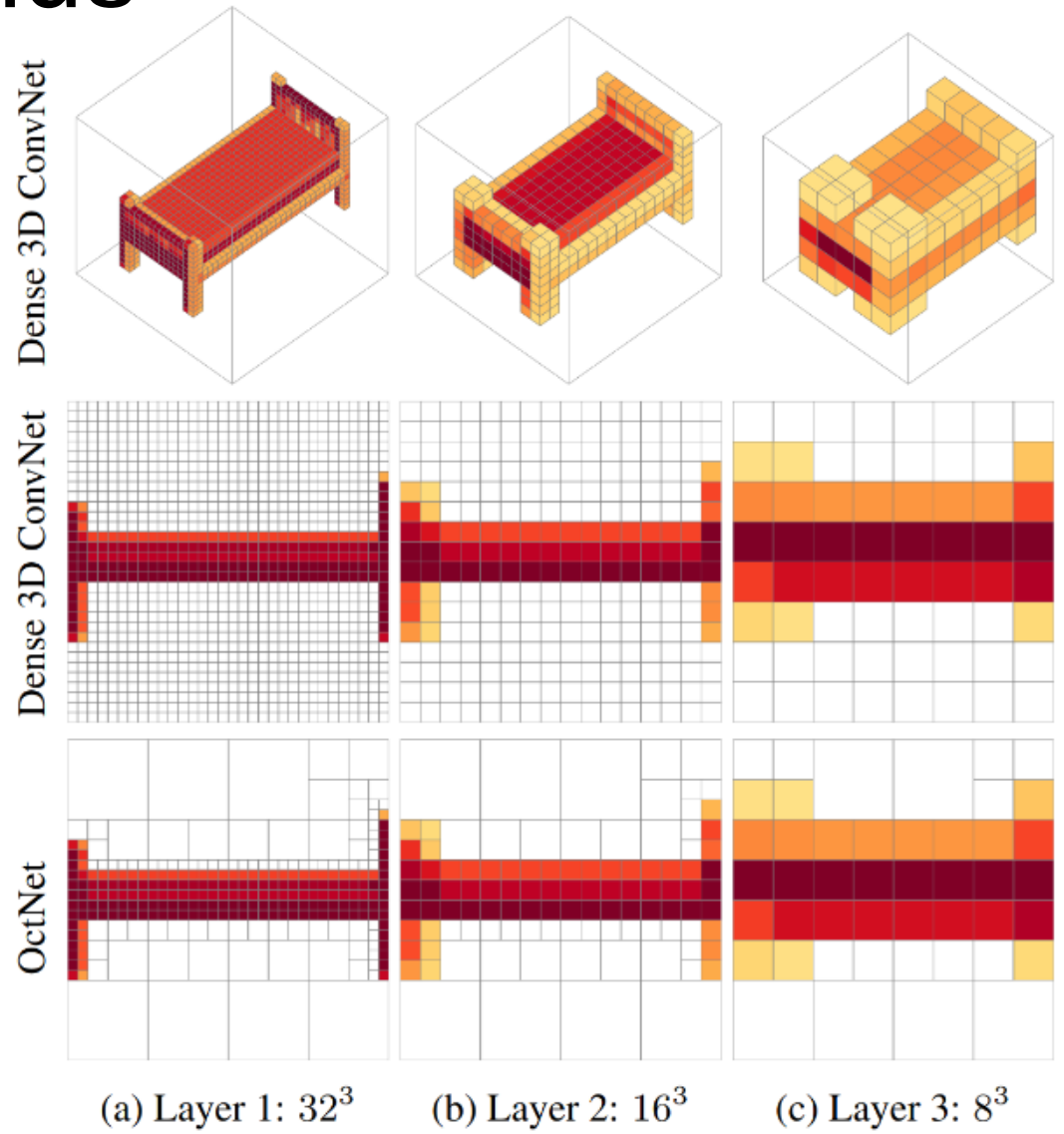


Deep Learning for Voxel Grids

OctNet

Adaptive grid resolution





Coding exercise

Write code that takes a point cloud as input and converts it to a voxel grid:
https://colab.research.google.com/drive/1ffucmXJYFs33Loh4wsKU2L7fpiI9bYWq?usp=sharing

Make a copy of this Google Colab and complete the missing parts.

The script provides helper functionalities to visualize the data.

Once you are done, share the link with me per email (dl4cv.eci24@gmail.com), using the **Share** button on the top right corner of Google Colab.

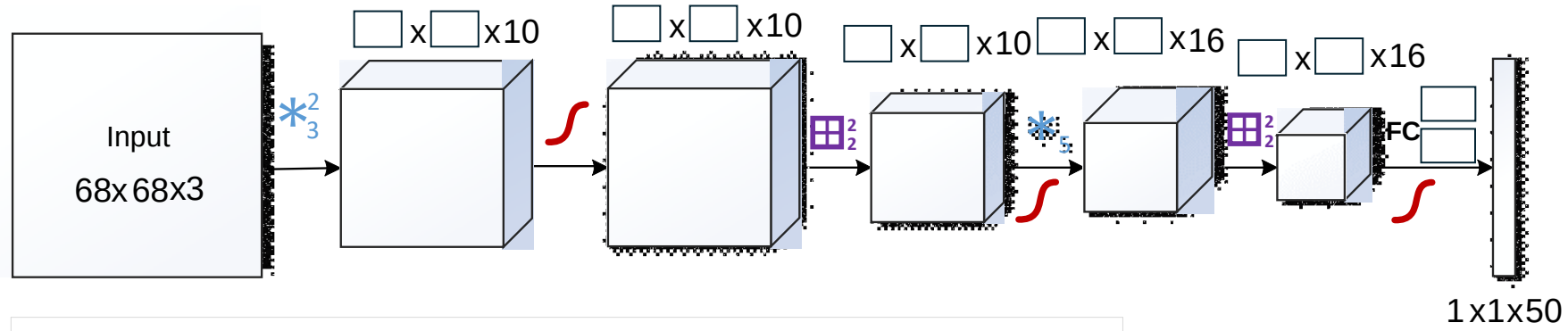
Run the code cells, and preferably let the running outputs there for me to see.

However, you can expect that I will try to run your coding cells myself to see if the output can be reproduced.

Exercise

No need to send the solution back to me, but you can ask questions!

Define the missing dimensions for the feature maps and layer weights below.
How does padding / no padding affect the result?



Basic layers:

Convolution	$*_n^{(s)}$	*s*- stride, *n*-kernel size
Pooling	$\boxplus_n^{(s)}$	
Activation	$\_/\ \int$	Fully-connected $FC_{output\_dim}^{input\_dim}$

Exercise

Rule:

Shape:

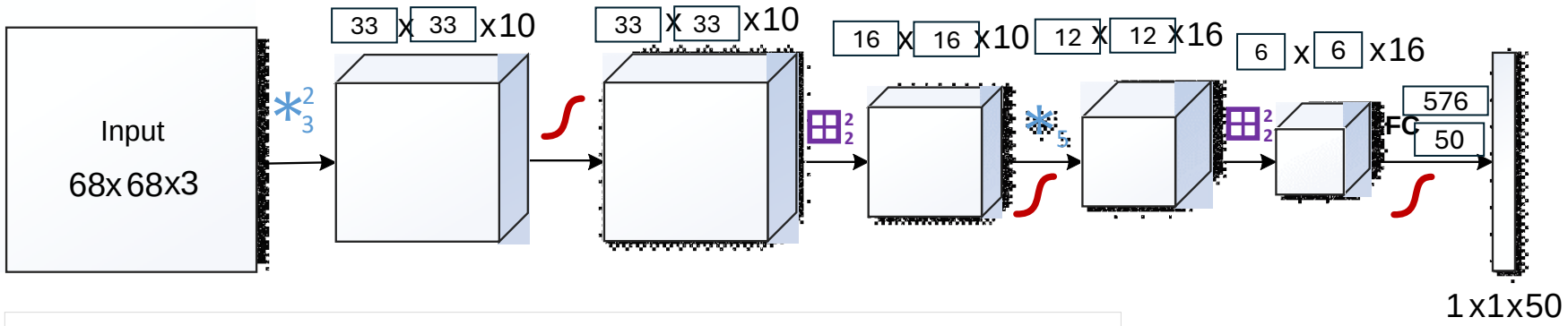
- Input: $(N, C_{in}, H_{in}, W_{in})$ or (C_{in}, H_{in}, W_{in})
- Output: $(N, C_{out}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$, where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

Solution for no padding



Basic layers:

Convolution	$*_n^{(s)}$	*s*- stride, *n*-kernel size
Pooling	$\boxplus_n^{(s)}$	
Activation	$\_/\ \int$	Fully-connected $FC_{output\_dim}^{input\_dim}$