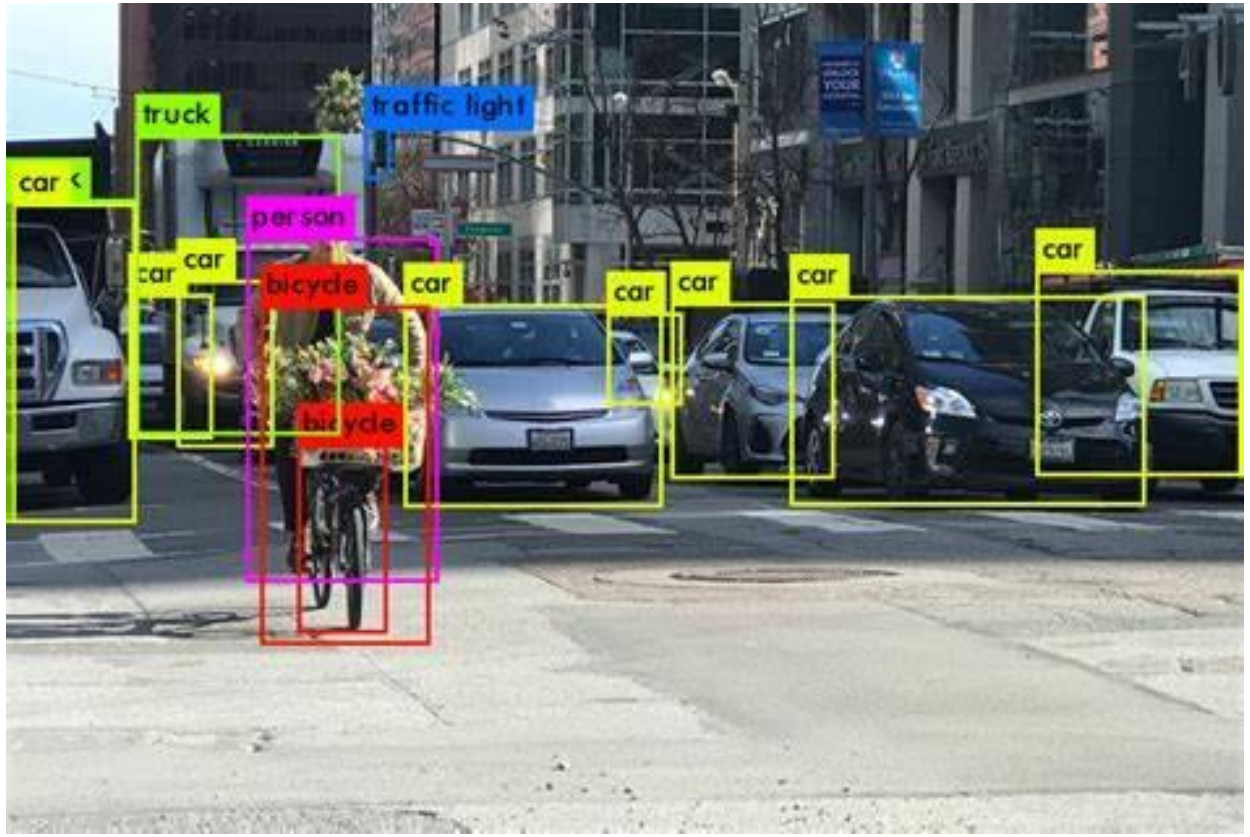
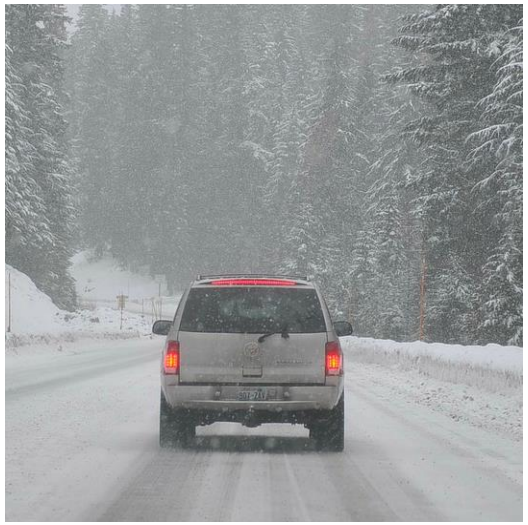
16. Detection & Segmentation & NeRF & 3D AIGC

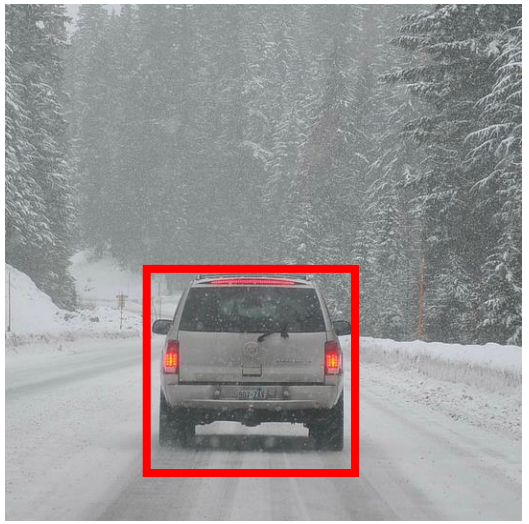


Localization and Detection

image classification



classification with localization

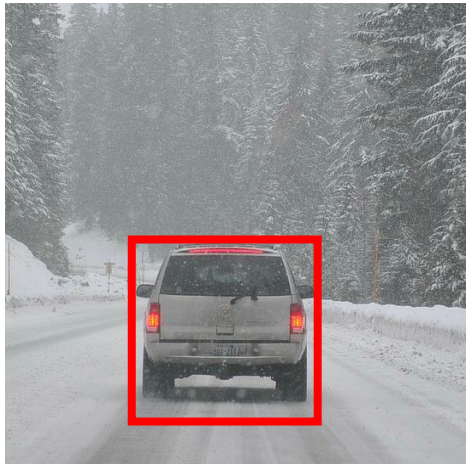
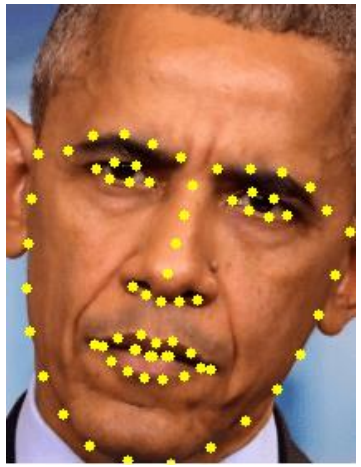


detection

one big object in a single image

multiple objects (even of different categories)

Classification with Localization

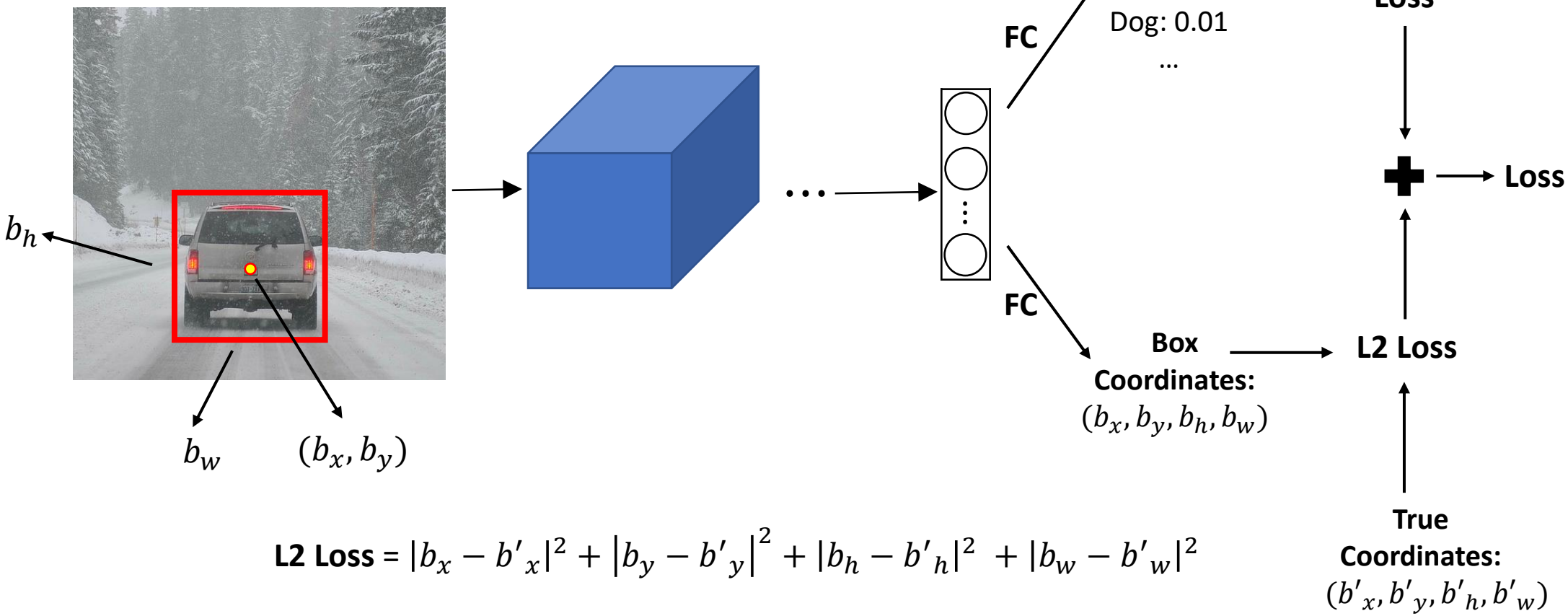




- Instead of just output class scores, output landmark x, y coordinates
- Manually mark out landmark positions in training data
  - The landmarks must be sorted consistently in all training data
- Use the Euclidean distance as the loss function

$$L(l, \hat{l}) = \sum_k |l_k(x) - \hat{l}_k(x)|^2 + |l_k(y) - \hat{l}_k(y)|^2$$

Classification with Localization

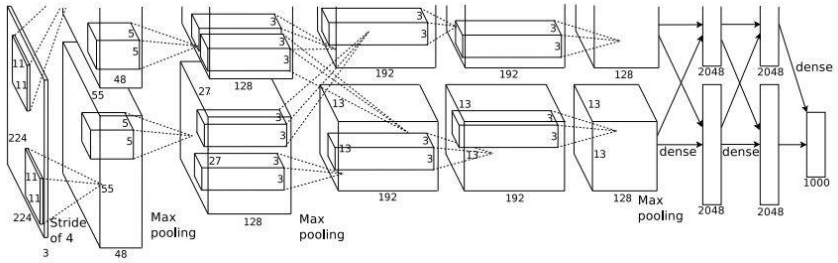
- Same in the landmark detection



L2 Loss = $|b_x - b'_x|^2 + |b_y - b'_y|^2 + |b_h - b'_h|^2 + |b_w - b'_w|^2$

Sliding Window Detection

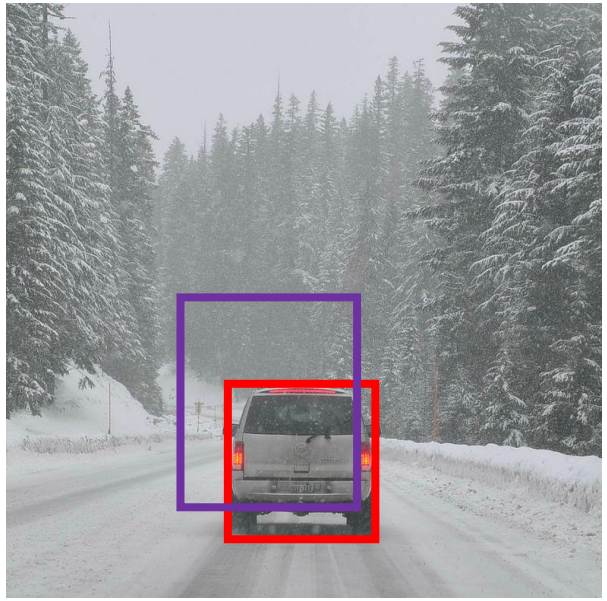
- Apply a CNN to many different crops of the image
- Classify each crop as object or background



Dog?	Yes
Cat?	No
Background?	No

IoU (Intersection over Union)

- Multiple detections of a single object
  - E.g. a crop with just the head of a dog might fire a detection
- Need a measure for the overlap between two bounding boxes.



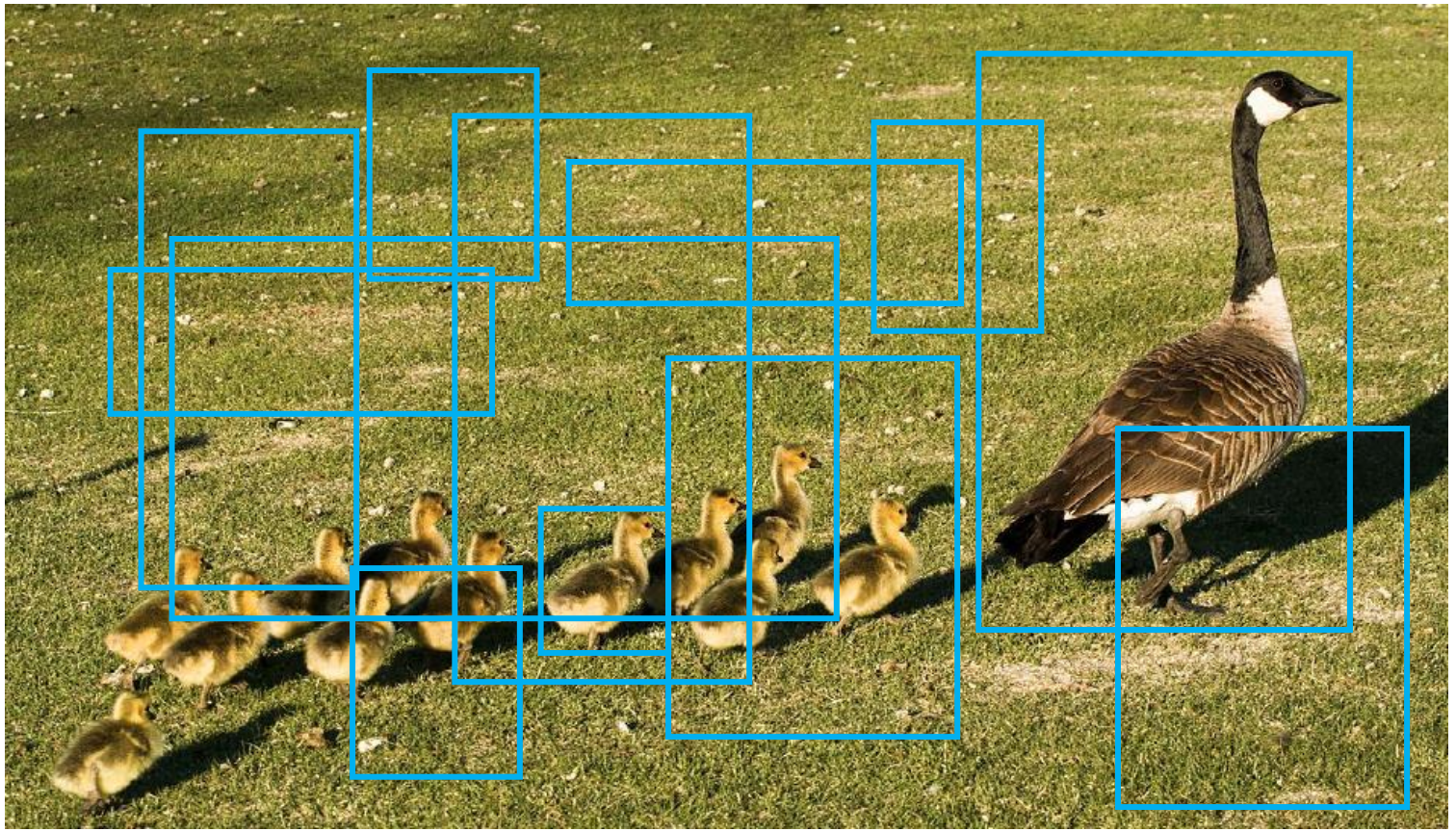
$$IoU = \frac{\text{area of } \square \cap \square}{\text{area of } \square \cup \square}$$

Non-maximum Suppression

- Pick the box with the largest score, output that as a prediction.
- Discard any remaining box with IoU ≥0.5 with the box output in the previous step

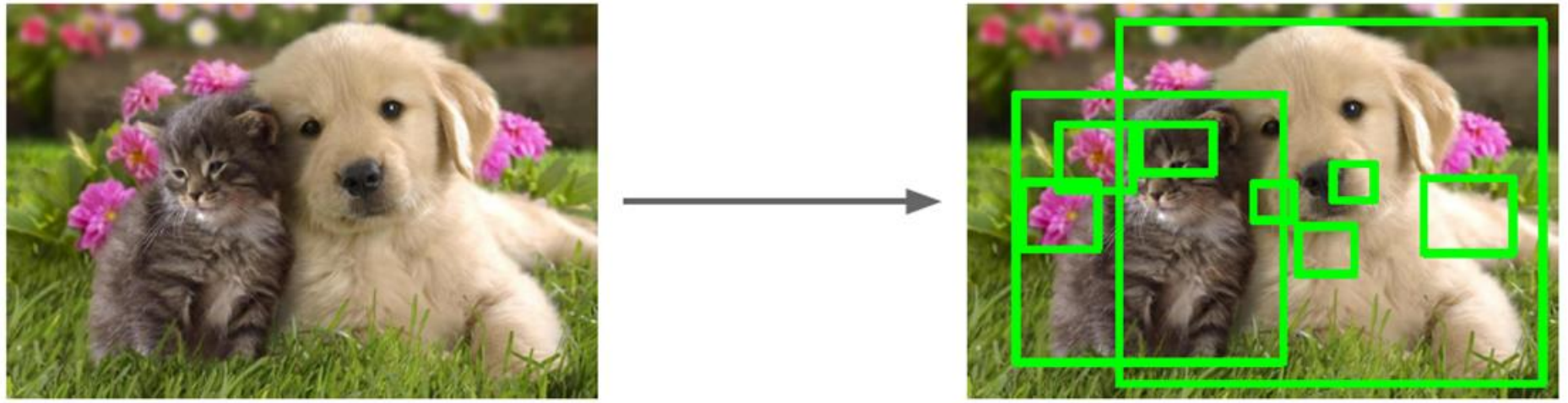
Sliding Window Detection

- Need many windows
  - Different position
  - Different scale
  - Different aspect ratio
- Enumerating (with quantization) all possible scale and aspect ratio is too slow



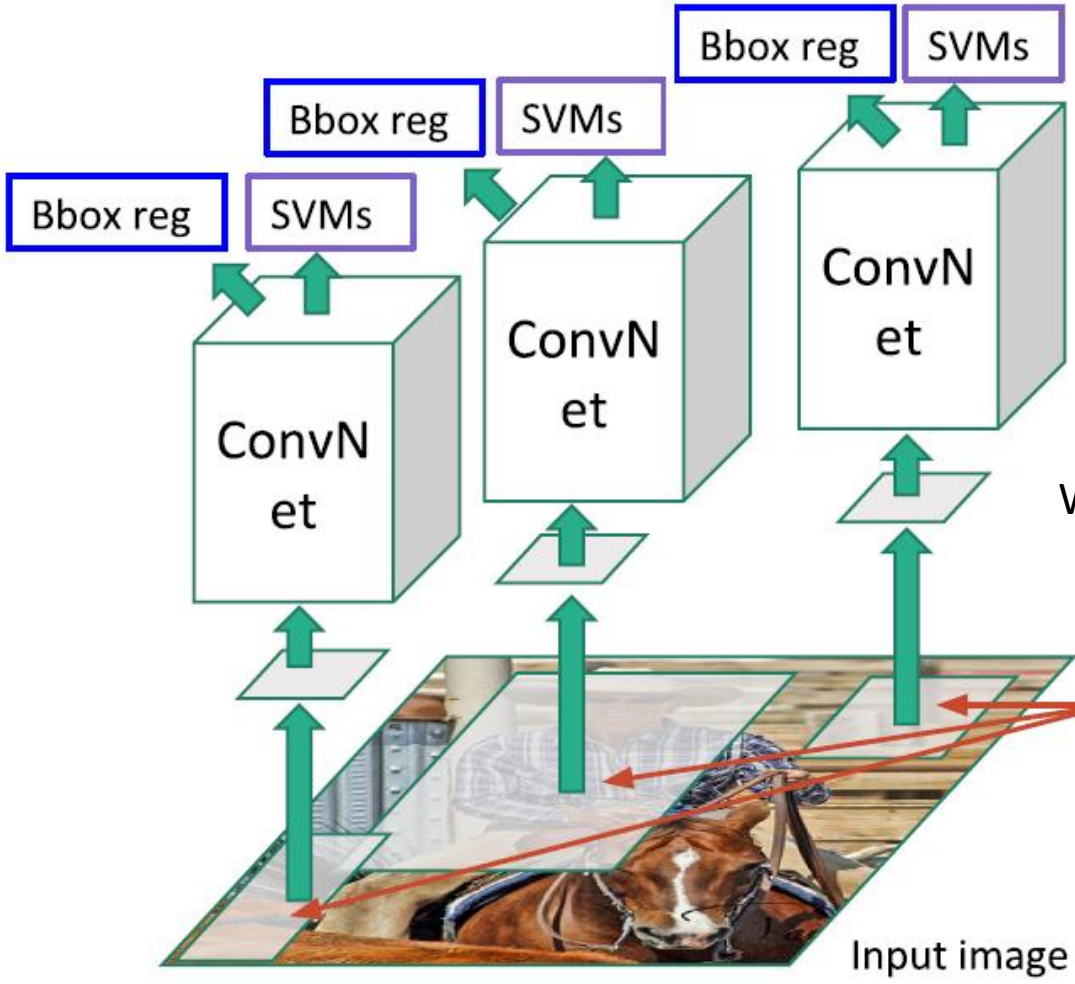
region proposals

- Find ‘blobby’ image regions that are likely to contain objects
- Relatively fast to run, e.g. the Selective Search gives 2k regions in a few seconds on a CPU



Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

R-CNN (Regions with CNN Features)



SVM classifier and
Bounding box regresser

Forward each region
through ConvNet

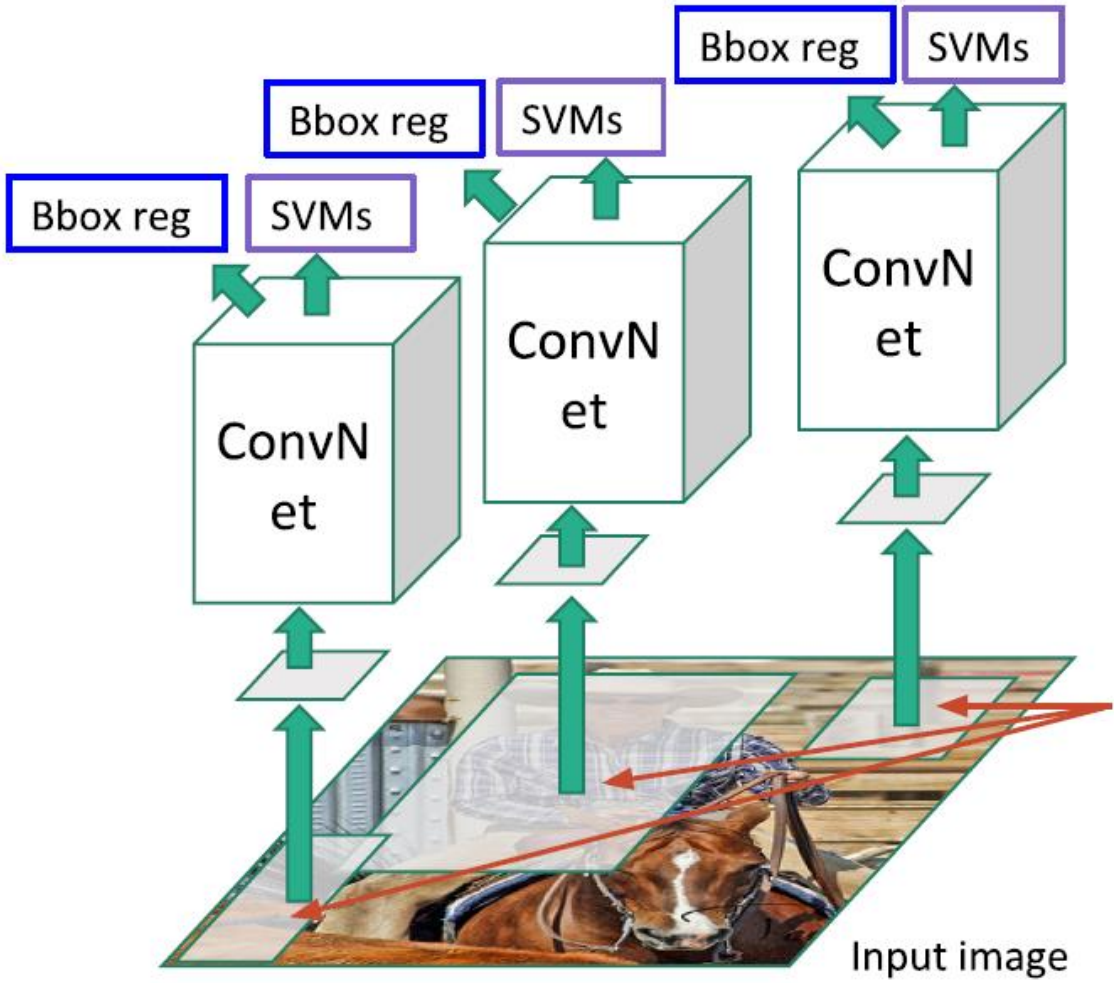
Warped image regions

Regions of Interest
(RoI) from a proposal
method (~2K)

Warping is needed because ConvNet takes a fixed size input, e.g. 224x224x3

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014

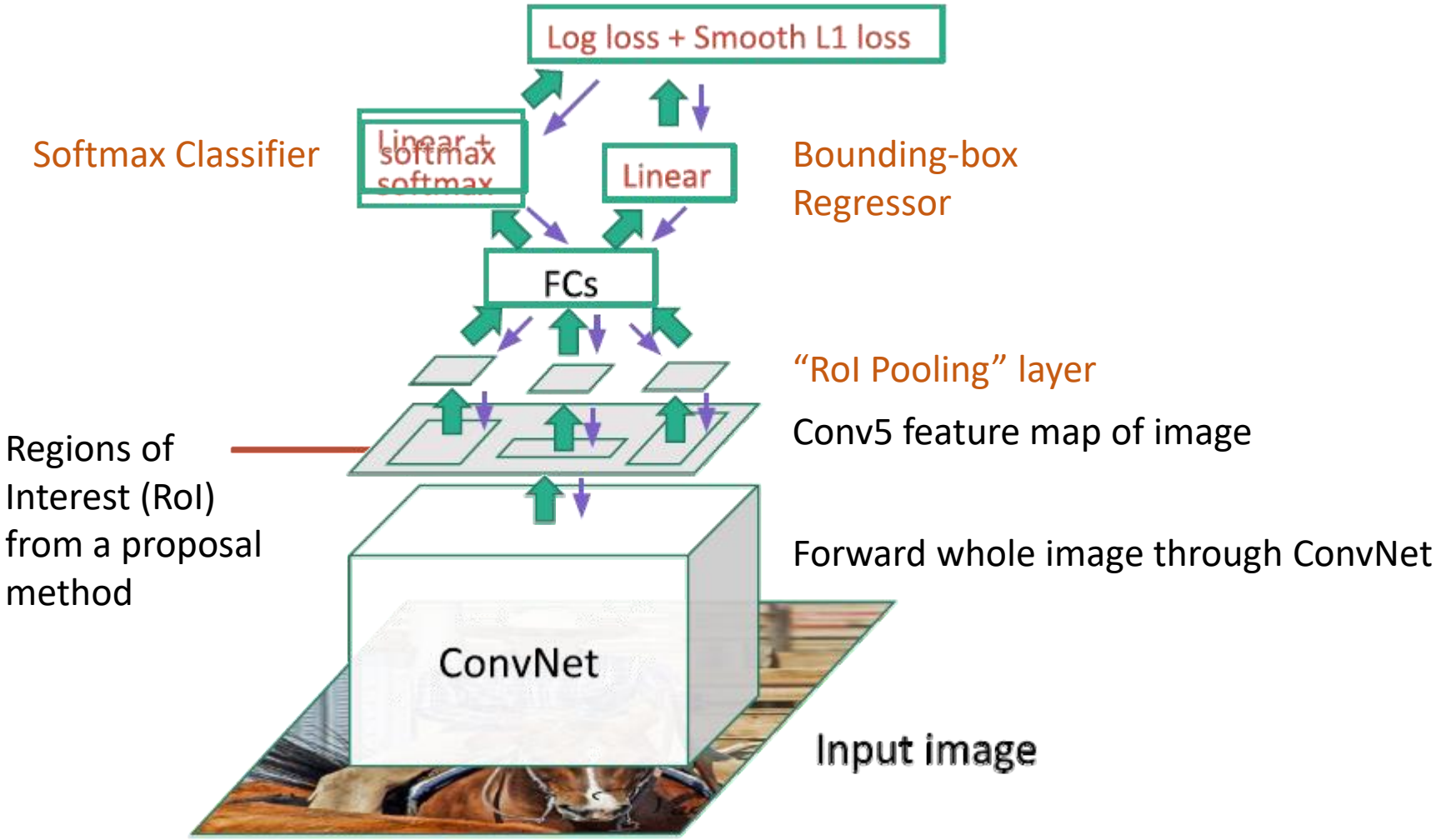
R-CNN (Regions with CNN Features)



- Going through the ConvNet for each proposal at testing time is slow
  - ~ 50 seconds per image
- Need a more efficient way
  - Reuse the computation at overlapping regions

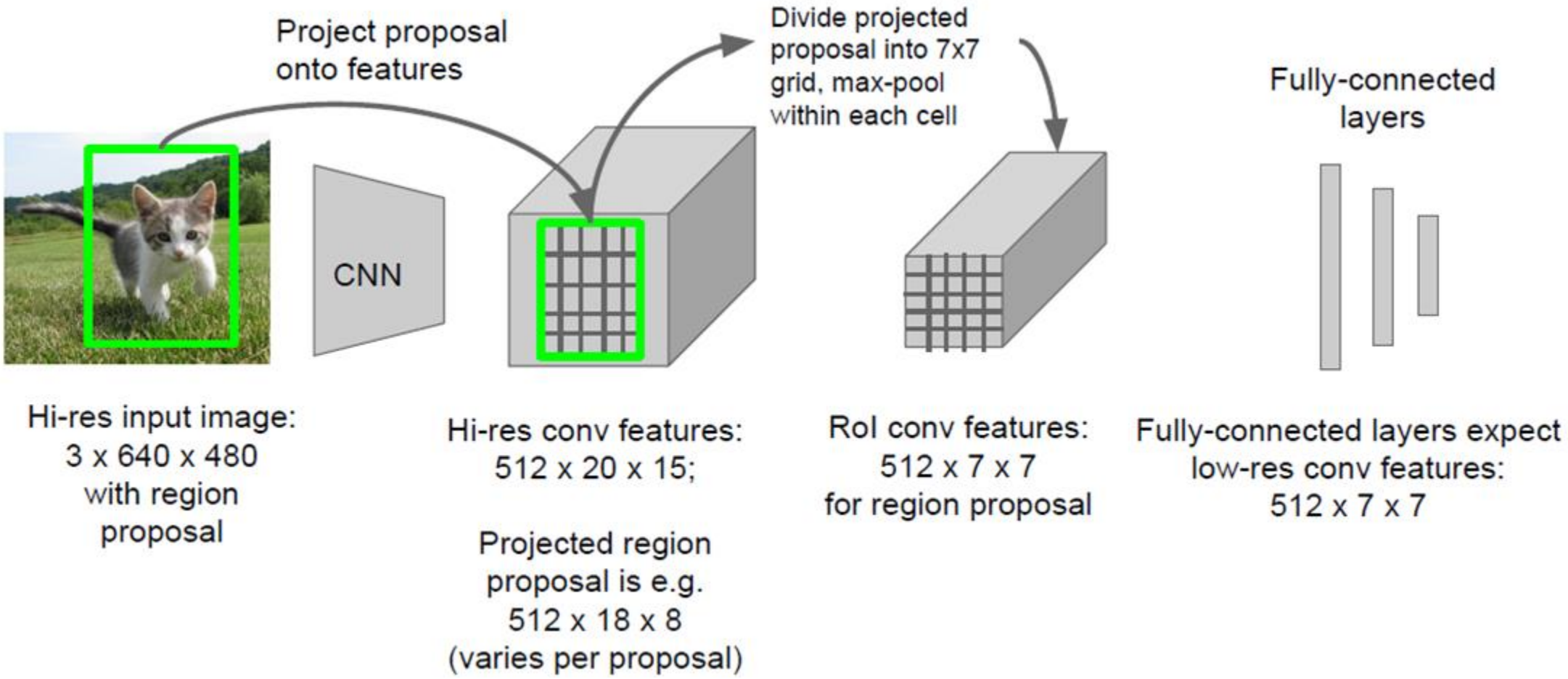
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014

Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015.

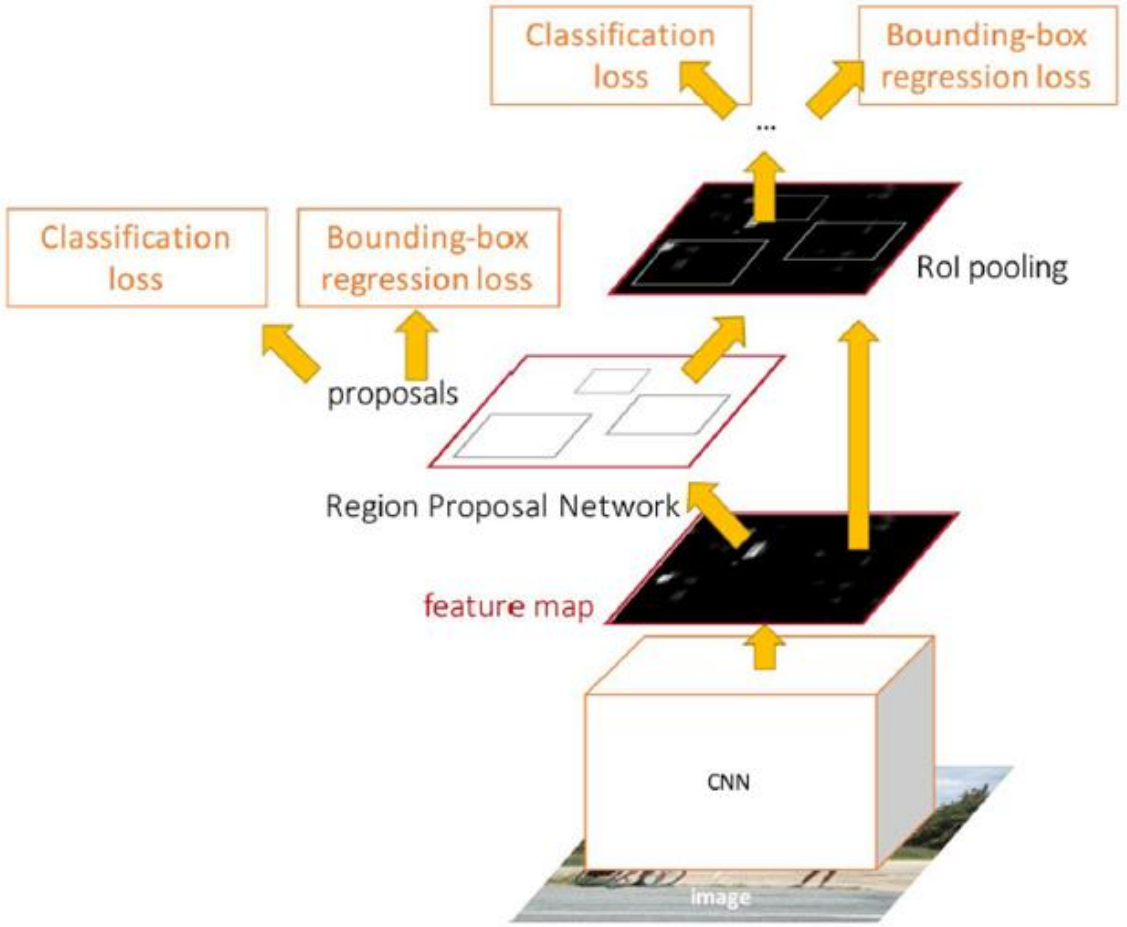
RoI Pooling



He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015.

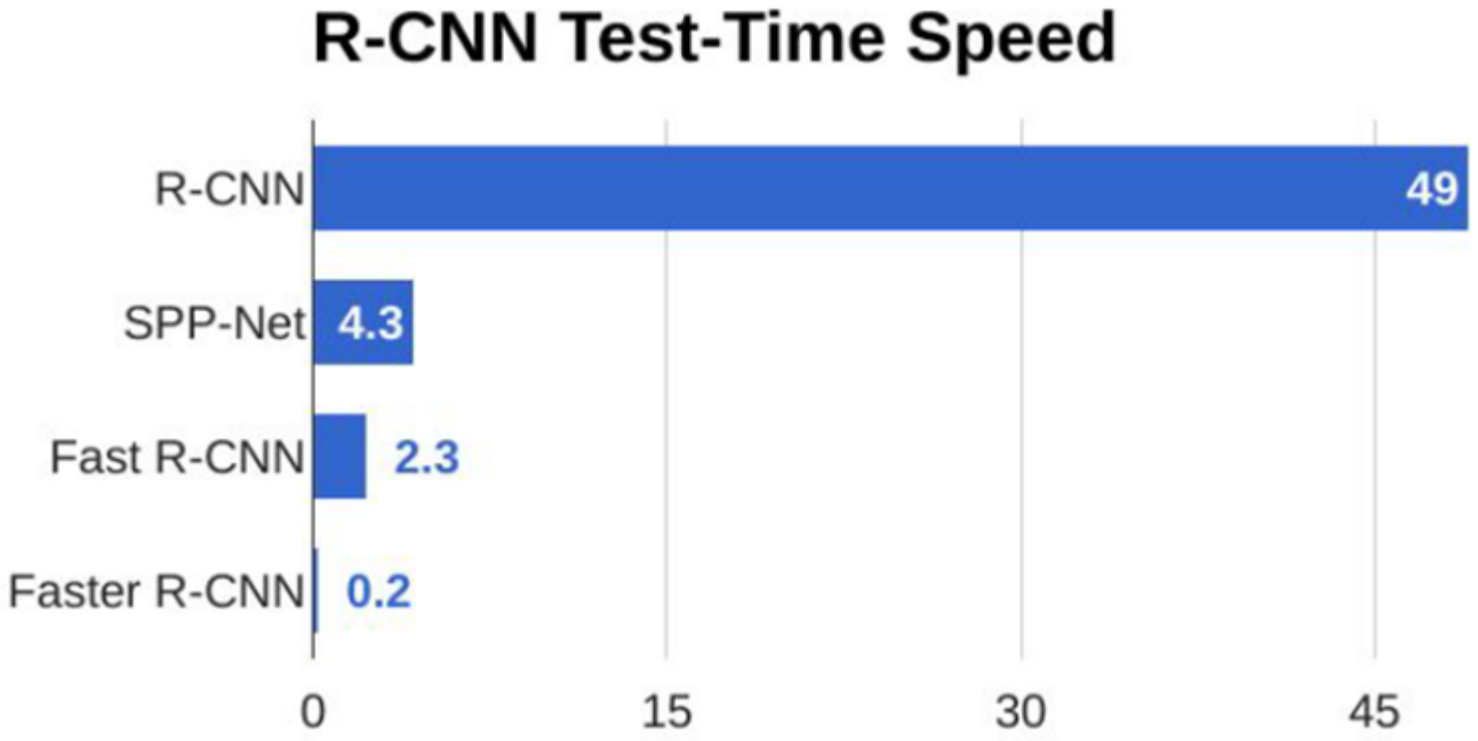
Faster R-CNN

- Region proposal is the bottleneck of Fast R-CNN
- Design a Region Proposal Network (RPN) to predict proposals
  - RPN share the same feature as the R-CNN (for classification and regression)
- Conv features are trained with 4 losses:
  - RPN classification of object or not object
  - RPN regression of box coordinates
  - Final classification score (softmax)
  - Final regression of box coordinates

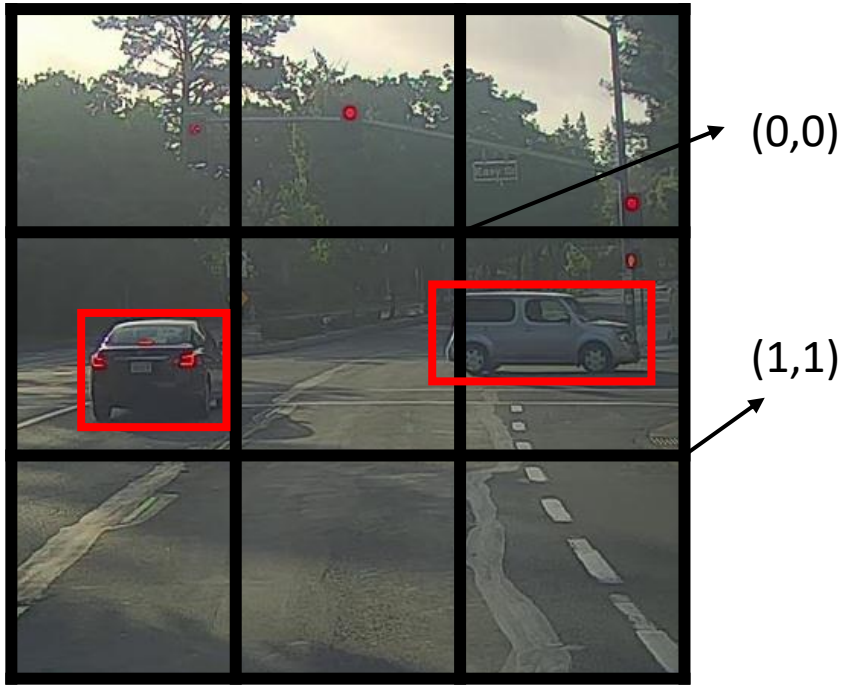


Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015

Faster R-CNN


R-CNN Test-Time Speed
R-CNN
SPP-Net
Fast R-CNN
Faster R-CNN
49
4.3
2.3
0.2
0
15
30
45

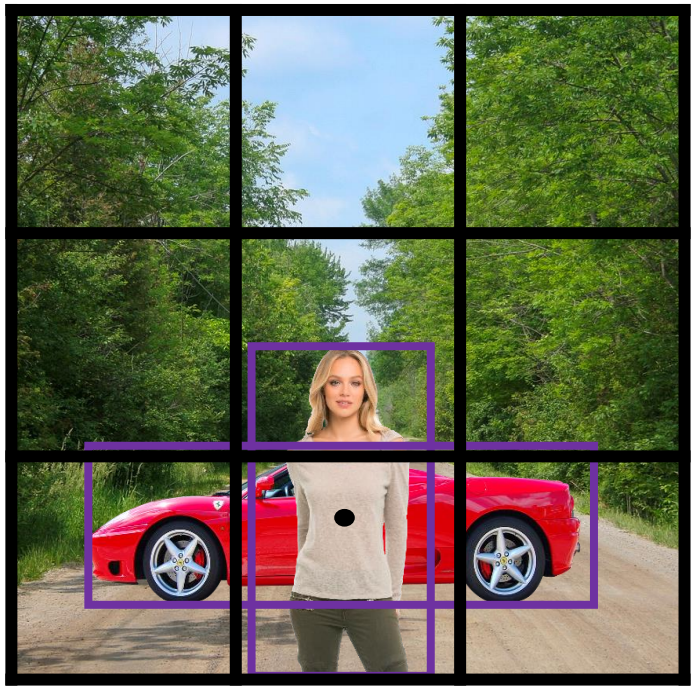
YOLO/SSD -- Detection without region proposals



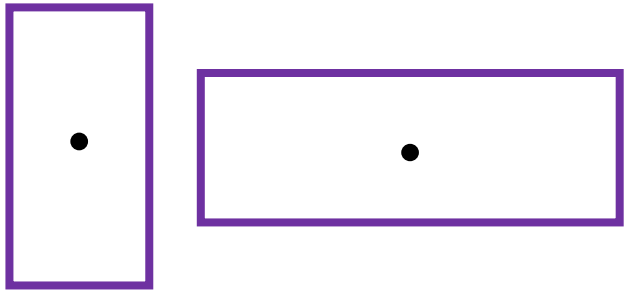
- Faster than Faster R-CNN (e.g. 50 fps vs 5 fps)
- Divide the image into grid cells, and regress 5+C numbers for each cell through a ConvNet
- Training:
  - Labels for training for each grid cell
    - $(p_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3, \dots)$; p_c is a confidence
    - b_x, b_y, b_h, b_w are defined wrt a cell; size can be > 1
  - Assign each object to the cell contain its center
  - Output volume: 3x3x(5+C)
    - In practice, a much finer grid is used, e.g. 19x19

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

anchor boxes



Anchor box 1: Anchor box 2:



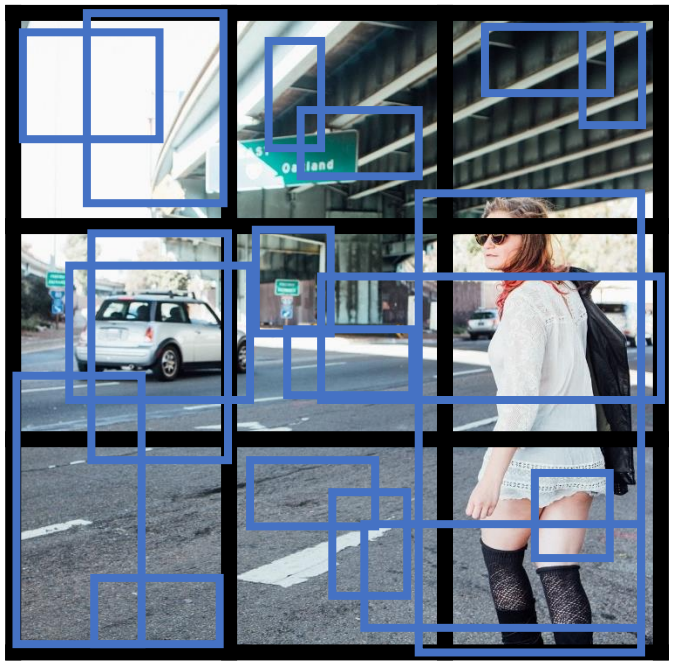
- To deal with multiple objects in a cell
  - Put multiple “anchor boxes” per cell
  - Regress 5 + C numbers per anchor box
- Training:
  - Assign an object to cell contains its center, and the anchor box with highest IoU
  - Output volume: 3x3x Kx(5+C)
- Choosing anchor boxes:
  - Often manually
  - Or choose the most representative boxes in training data (e.g. K-Means)

$$y = \begin{bmatrix} b_x \\ b_y \\ b_h \\ b_w \\ p_c \\ c_1 \\ c_2 \\ c_3 \\ b_x \\ b_y \\ b_h \\ b_w \\ p_c \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1

Anchor box 2

example: the non-max suppressed outputs

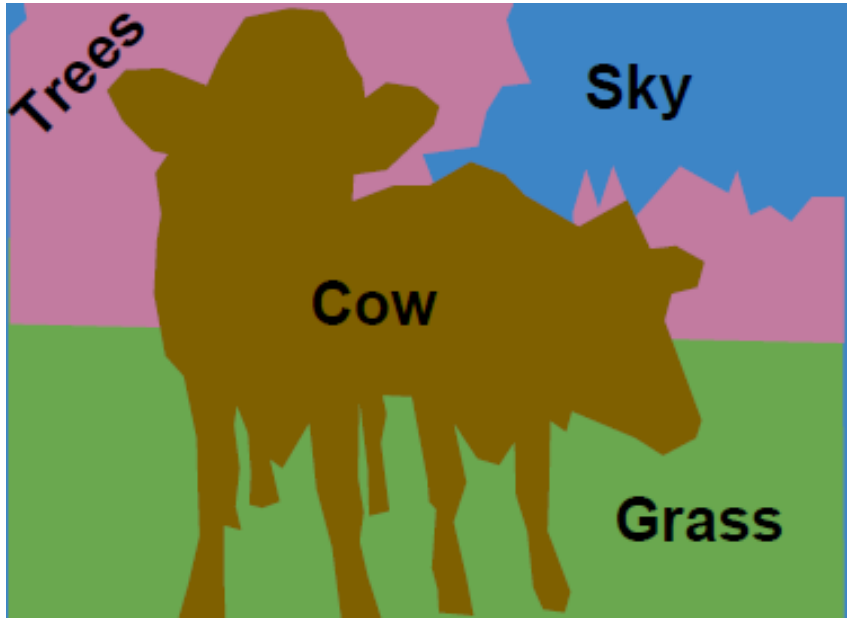


- For each grid call, get 2 predicted bounding boxes.
- Get rid of low probability predictions.
- For each class (e.g. pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

Questions?

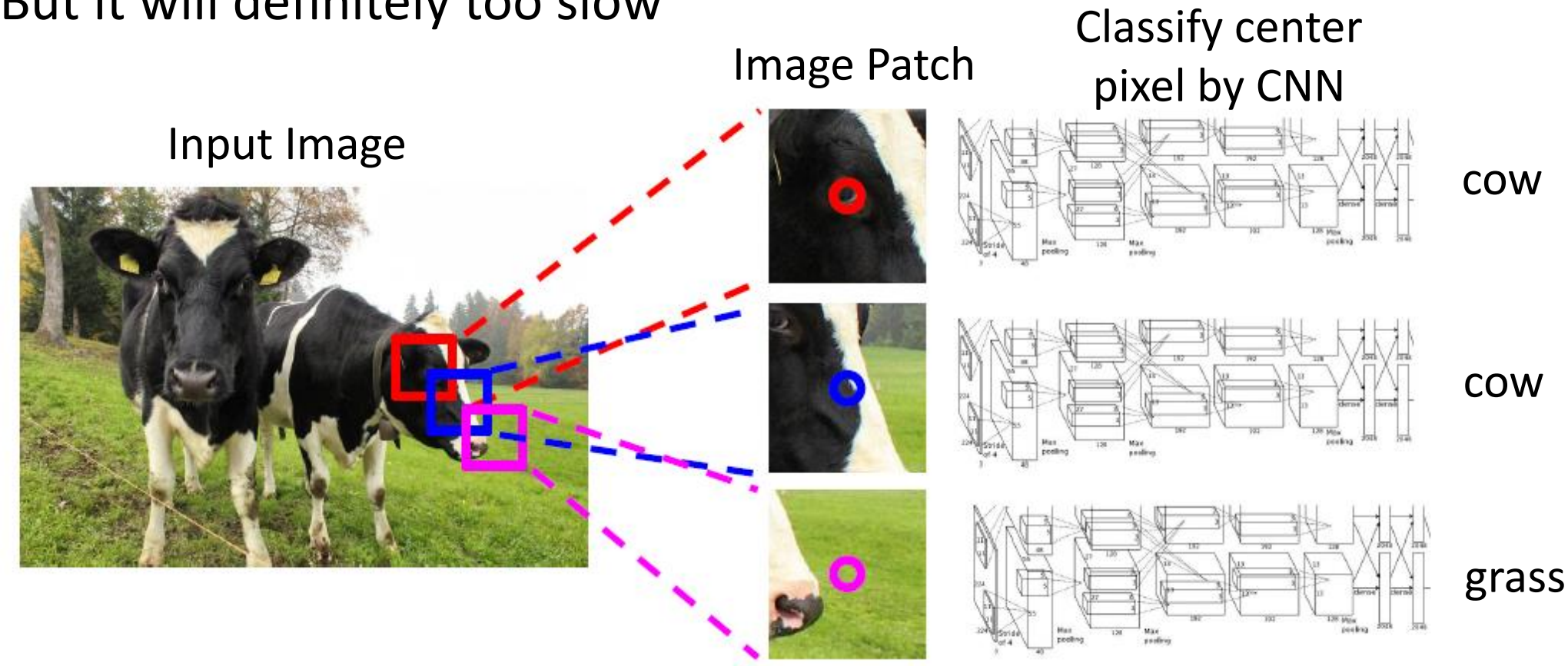
Semantic Segmentation

- Label each pixel with a category label
- Don’t differentiate instances, only care about pixels



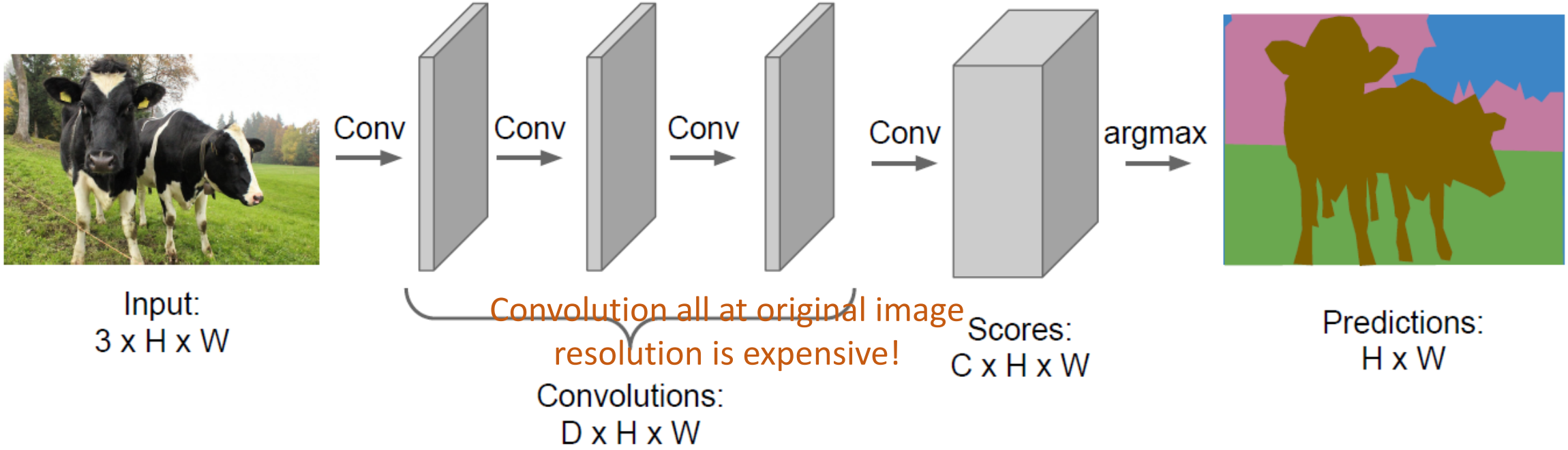
Semantic Segmentation

- Sliding window approach might be applied in principle
- But it will definitely too slow



FCN (Fully Convolutional Networks)

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



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

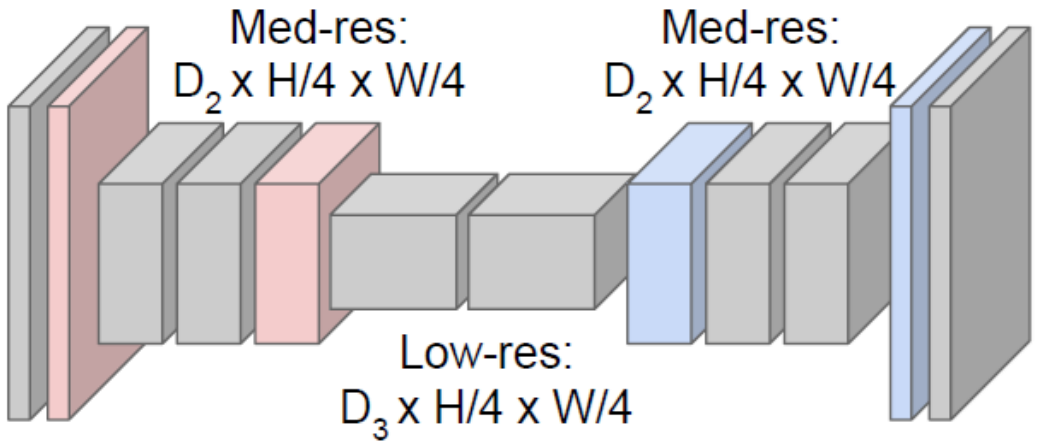
FCN (Fully Convolutional Networks)

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

Downsampling:
Pooling, strided convolution

Upsampling:
???

Input:
3 x H x W

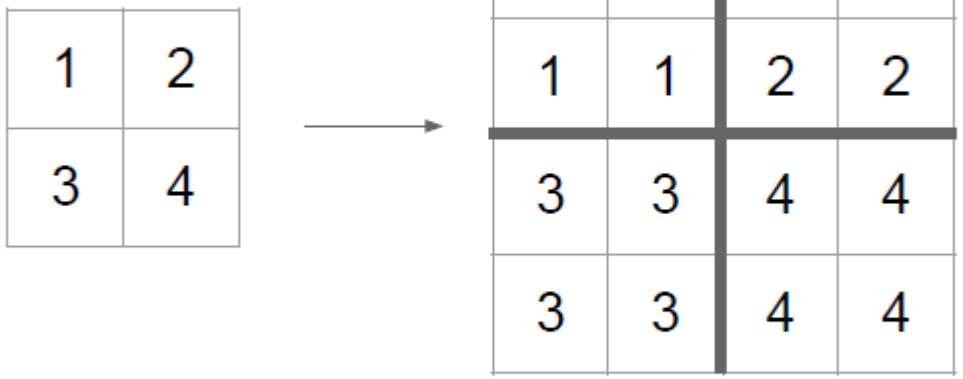


Predictions:
H x W

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

Network Upsampling: “Unpooling”

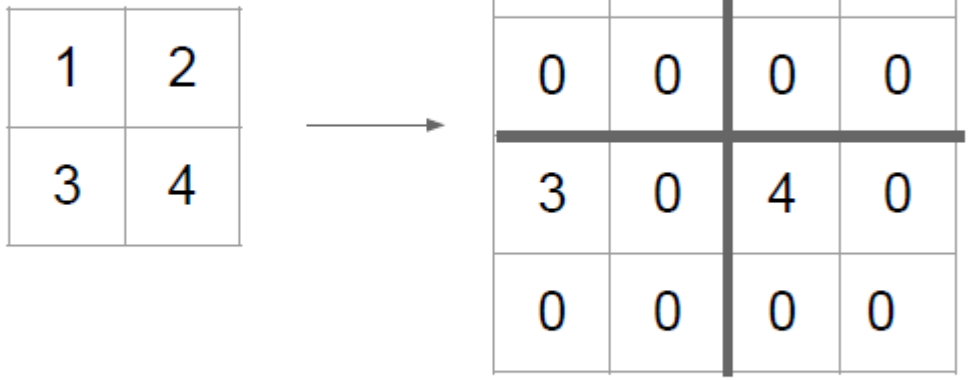
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”



Input: 2 x 2

Output: 4 x 4

Network Upsampling: “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

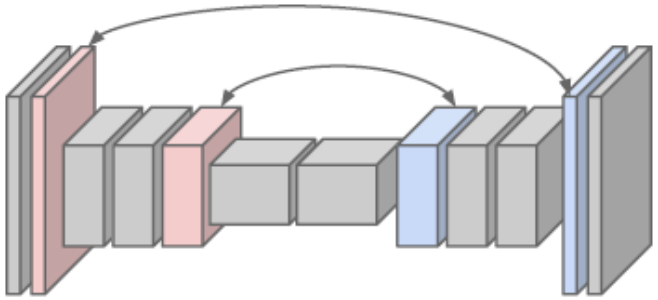
1	2
3	4

Input: 2 x 2

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

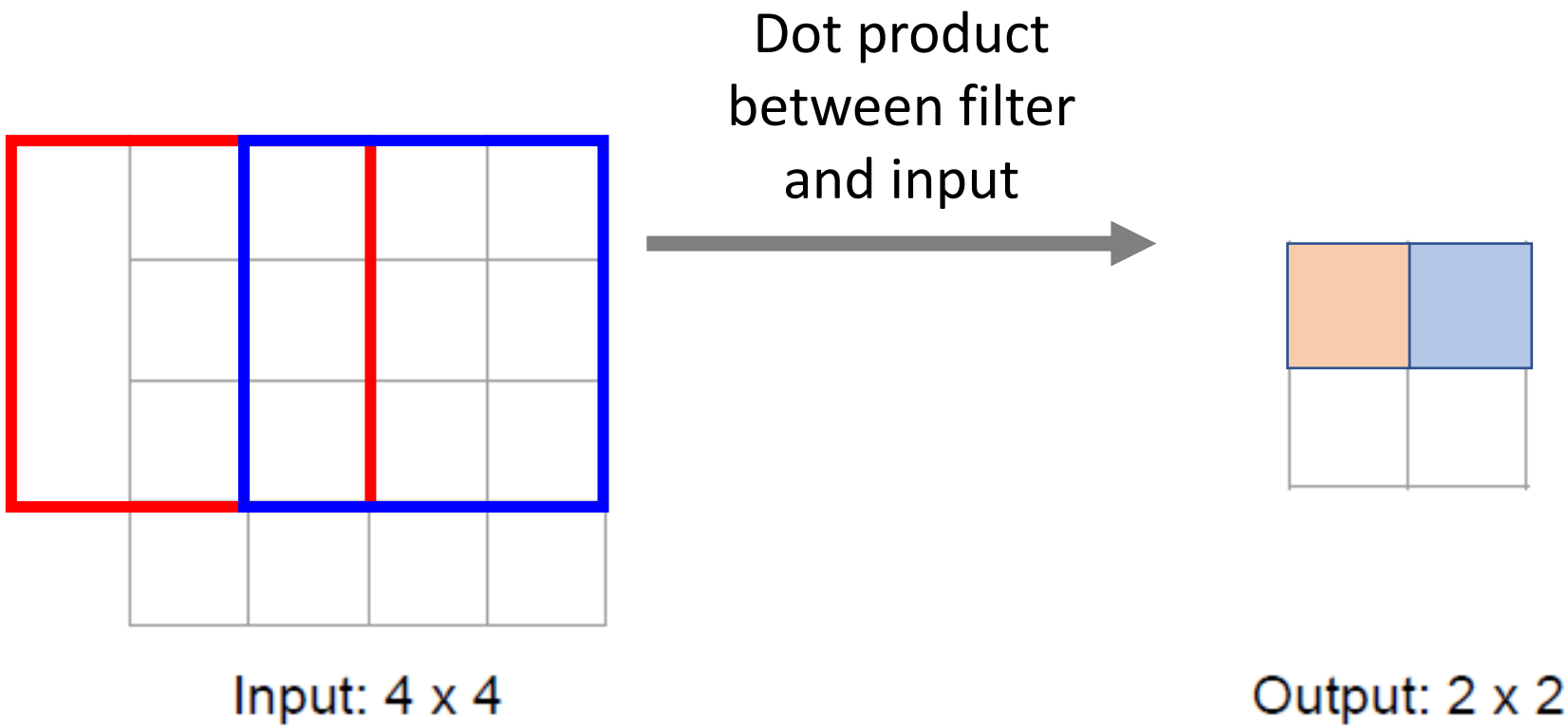
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



Learnable Upsampling: Transpose Convolution

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

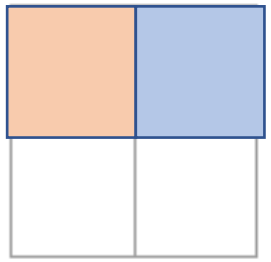


Learnable Upsampling: Transpose Convolution

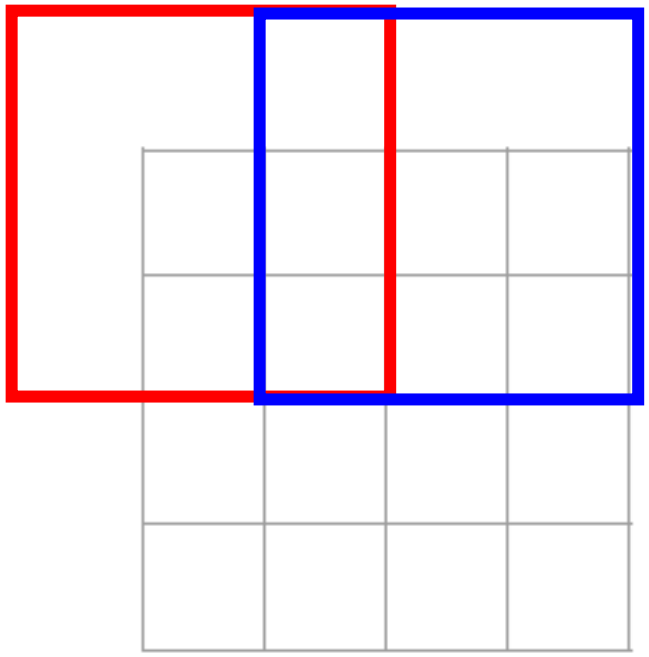
Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

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

Input gives weight for filter



Input: 2 x 2

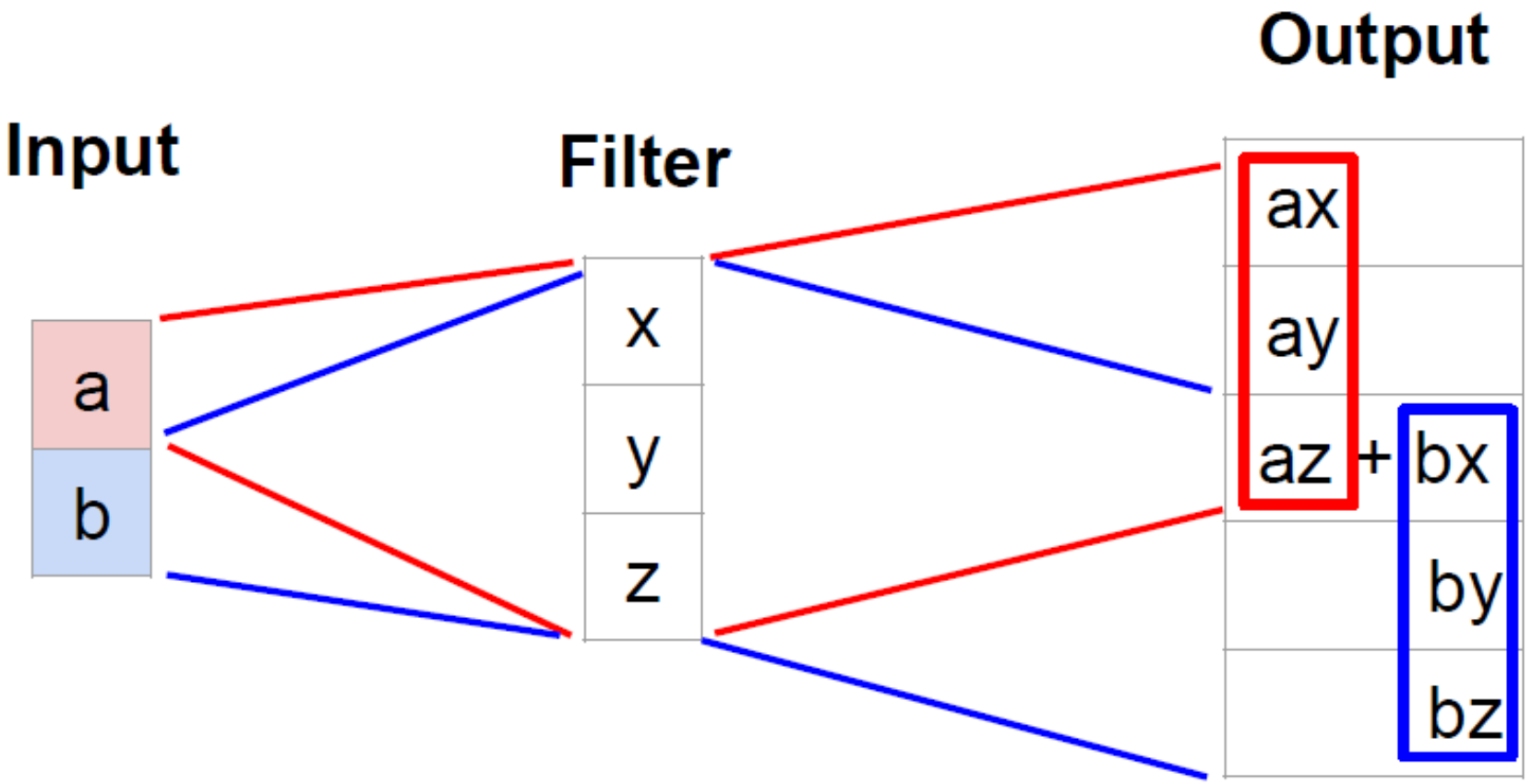


Output: 4 x 4

Sum where output overlaps

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

Learnable Upsampling: 1D Example

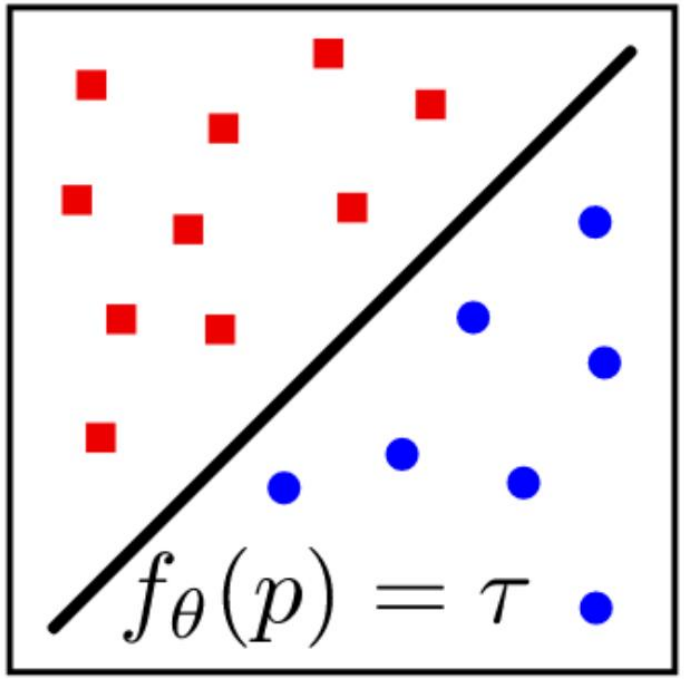


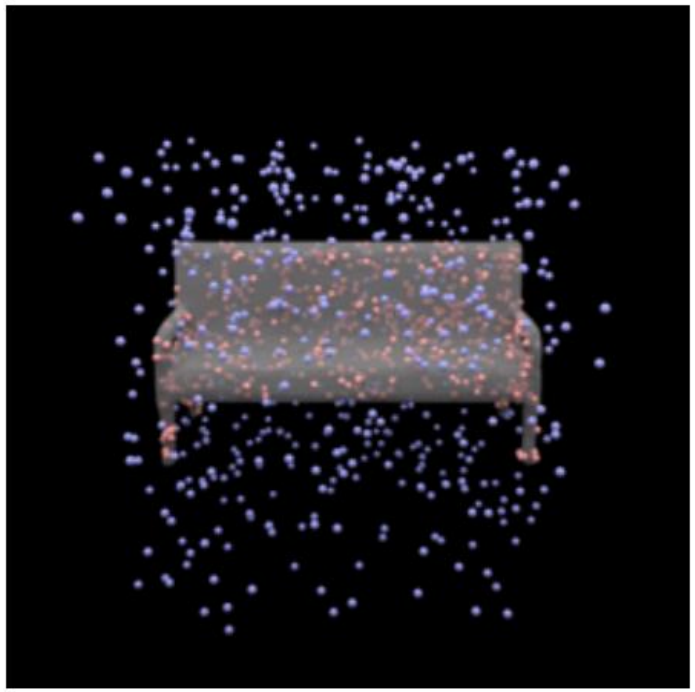
Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

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

Questions?

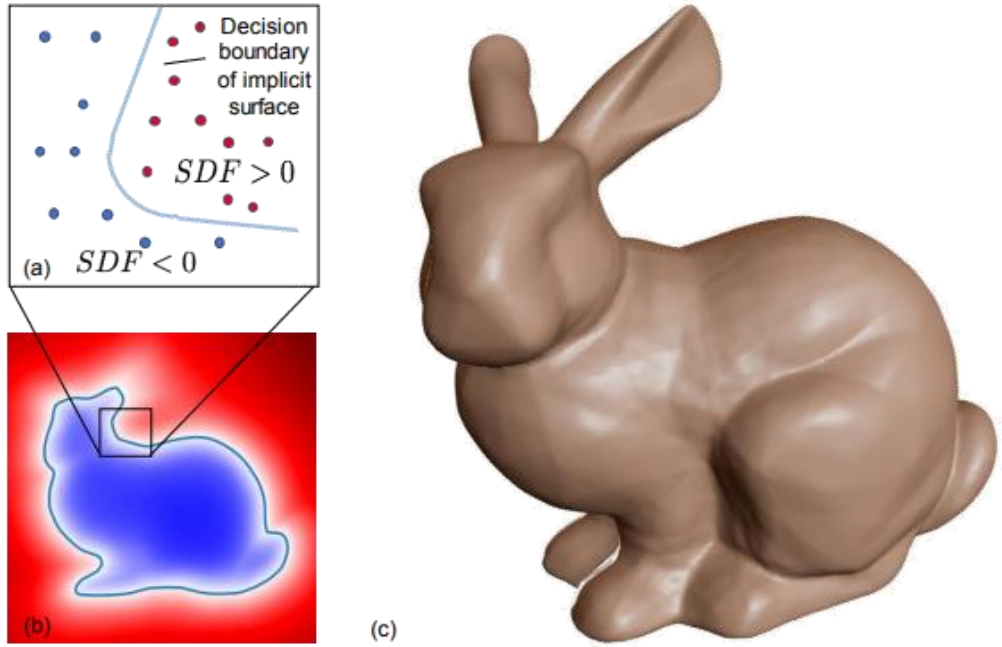
Neural networks as a continuous shape representation

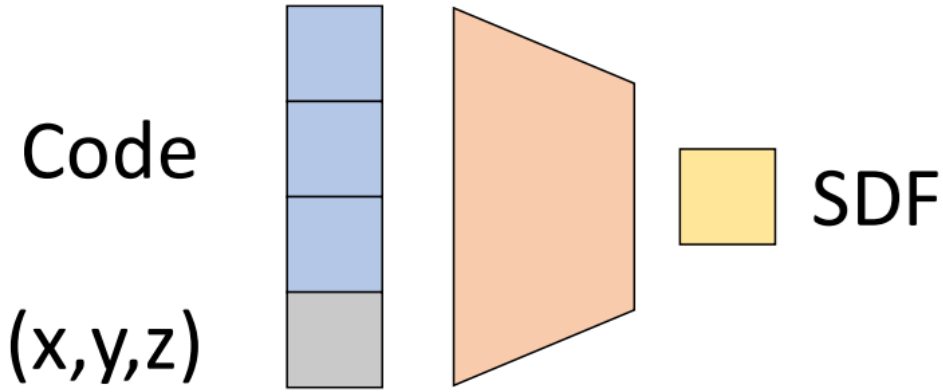




Occupancy Networks, Mescheder et al. CVPR 2019

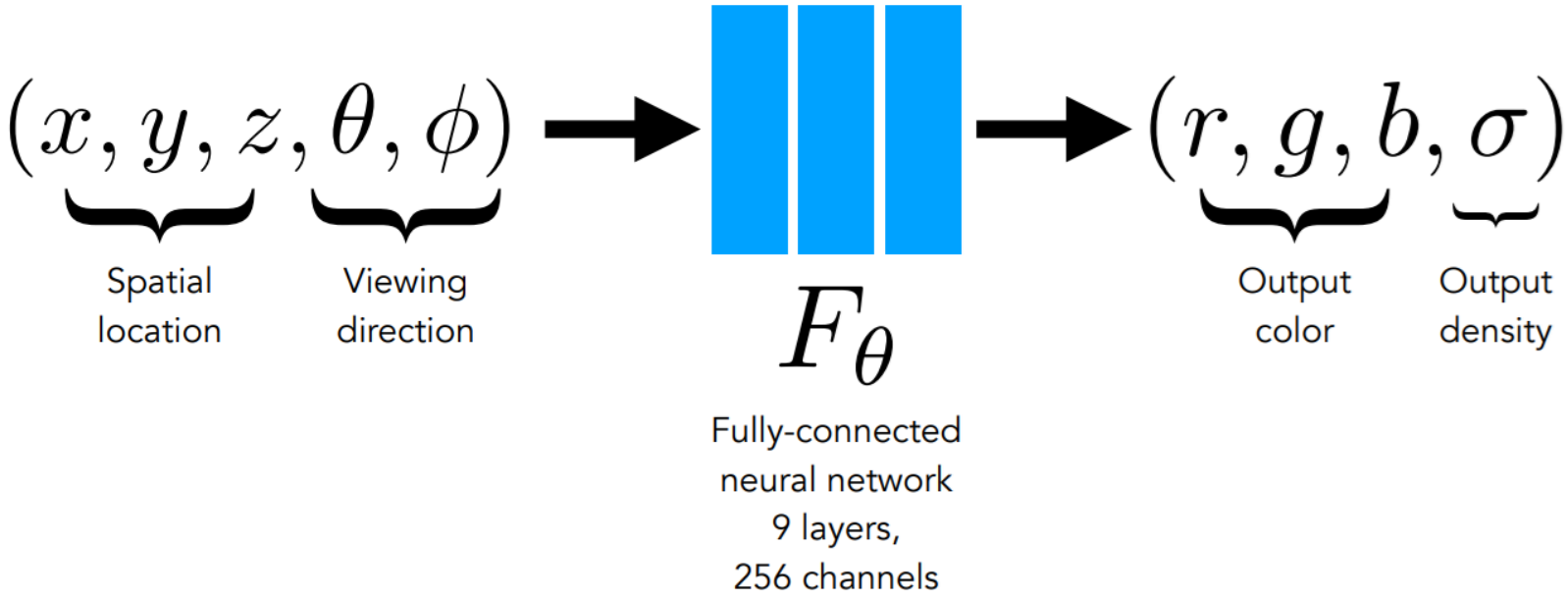
Neural networks as a continuous shape representation



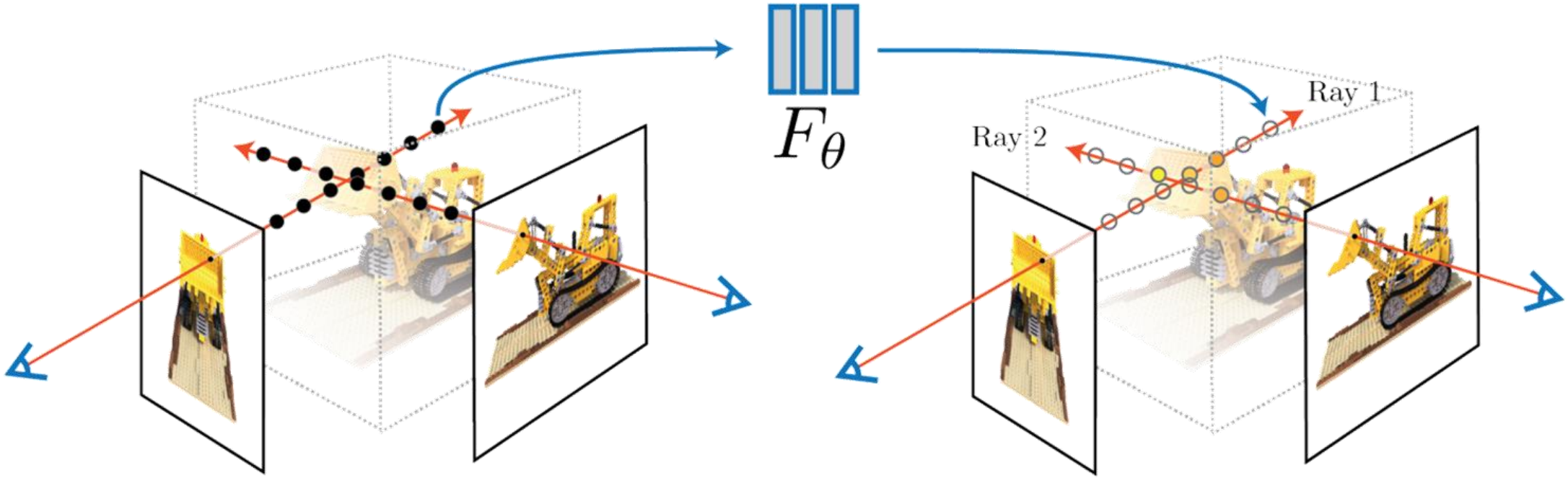


DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, Park et al. CVPR 2019

NeRF: Neural Radiance Fields



Generate views with traditional volume rendering



Volume rendering is trivially differentiable

- Rendering model for ray r(t) = o + td:

$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

colors

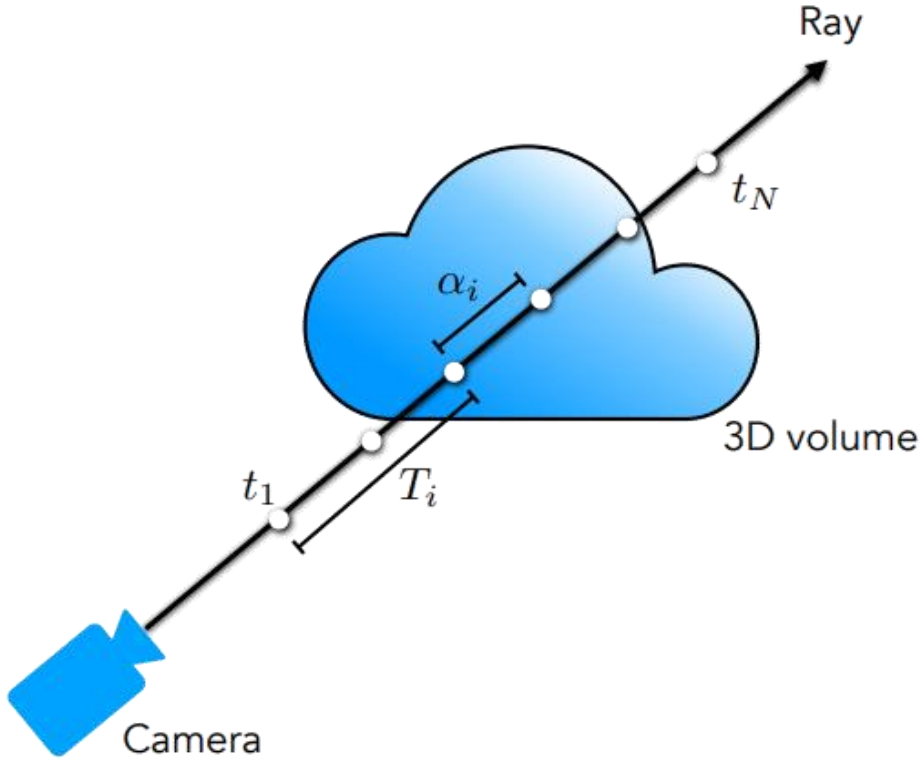
weights

- How much light is blocked earlier along ray:

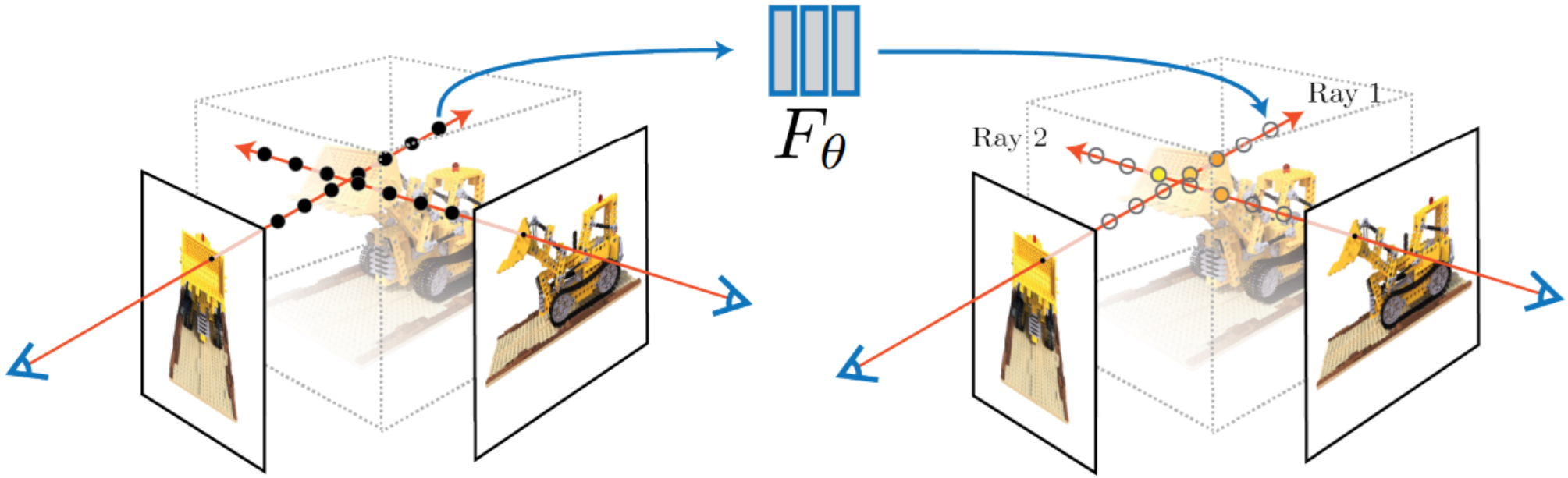
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

- How much light is contributed by ray segment i:

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$



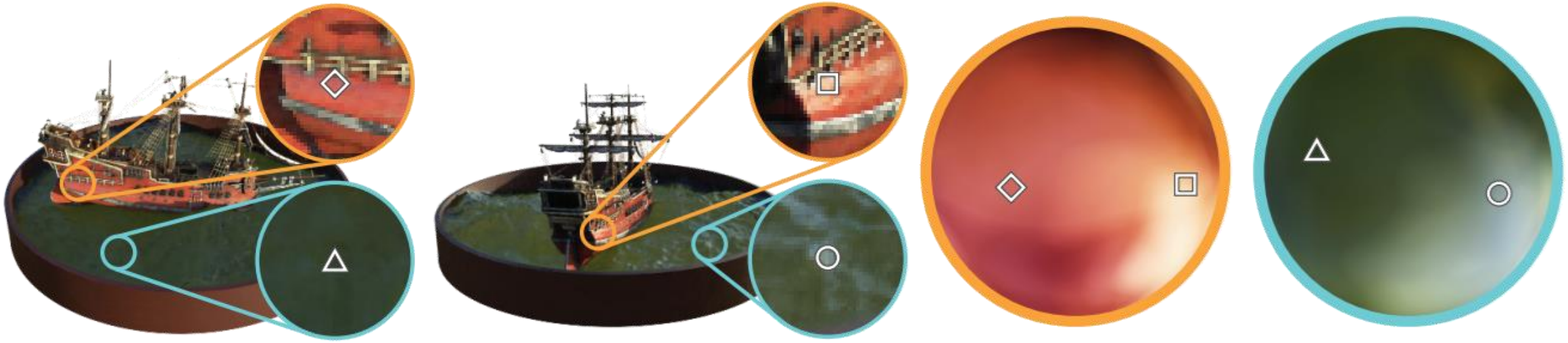
Optimize with gradient descent on rendering loss



$$\min_{\theta} \sum_{i} || \text{render}_i(F_\theta) - I_i ||^2$$

Training network to reproduce all input views of the scene

Viewing directions as input

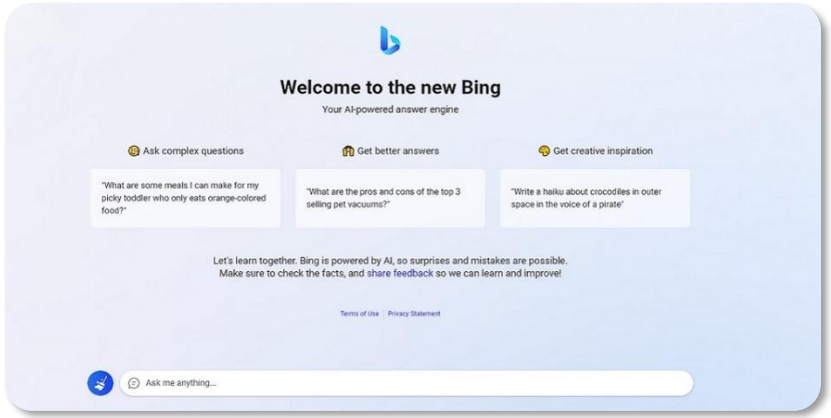


(a) View 1 (b) View 2 (c) Radiance Distributions

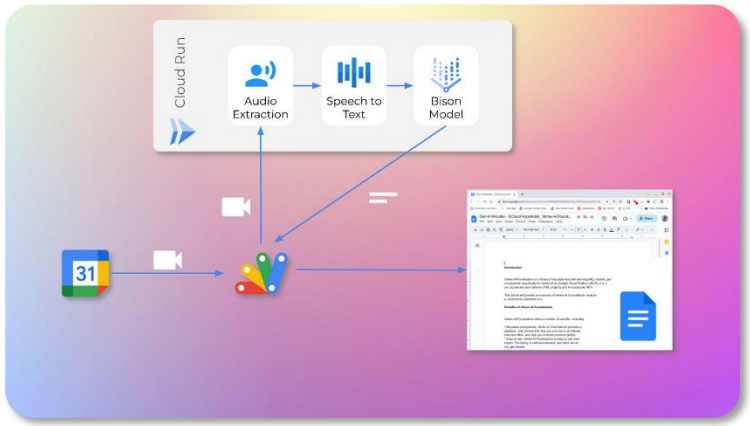
NeRF

Questions?

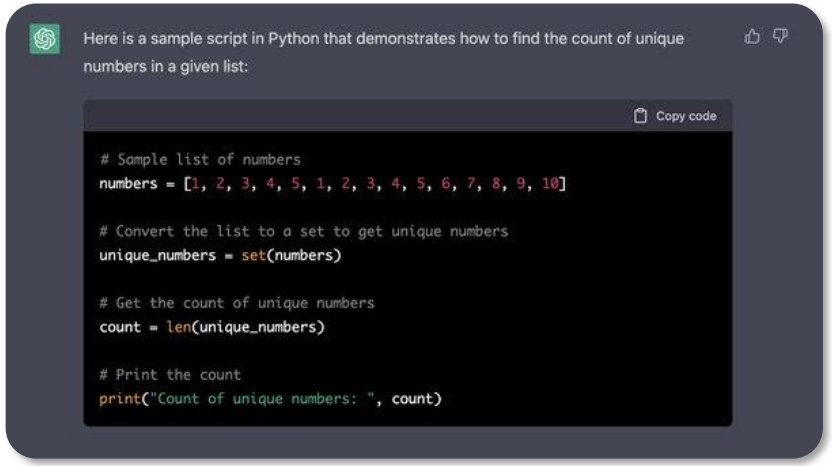
Text content generation becomes commercialized



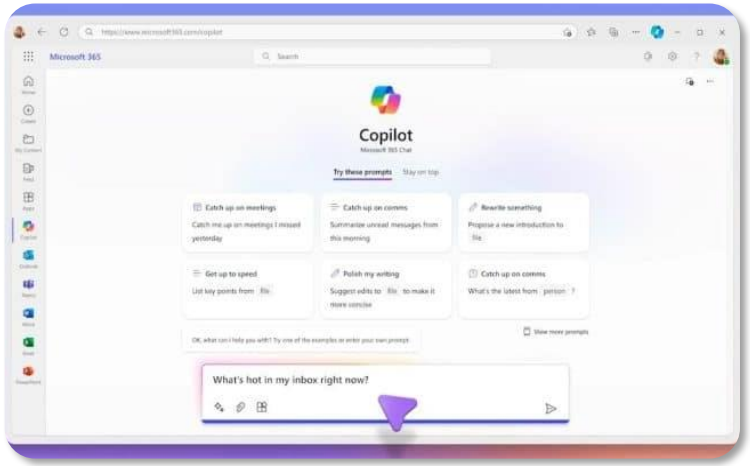
Search Engine



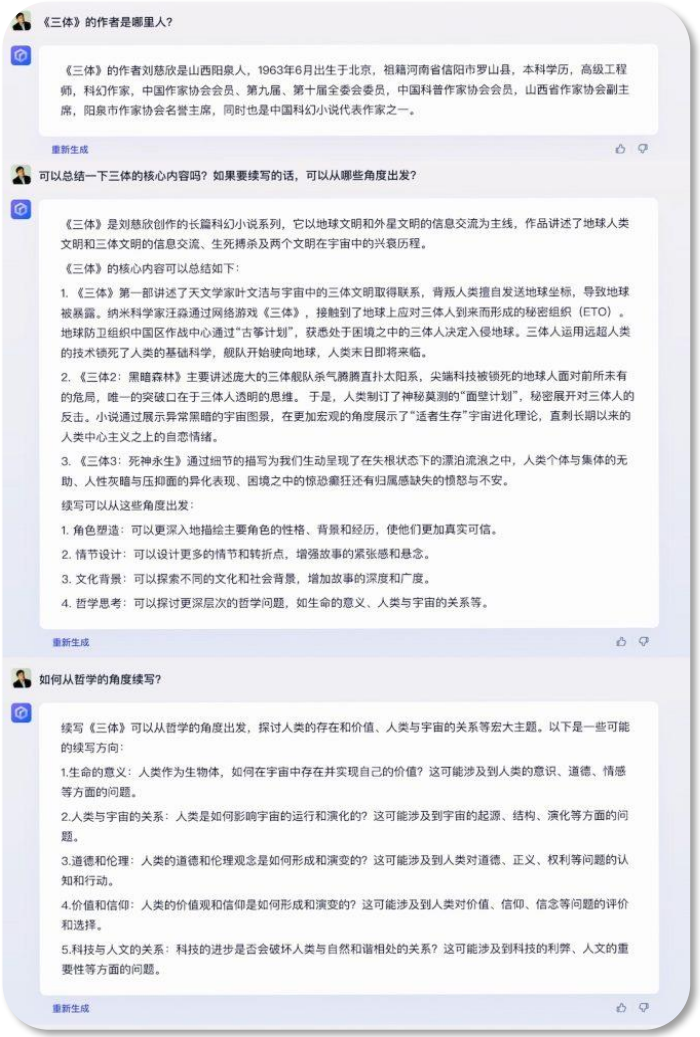
Meeting Record



Coding



Office Software

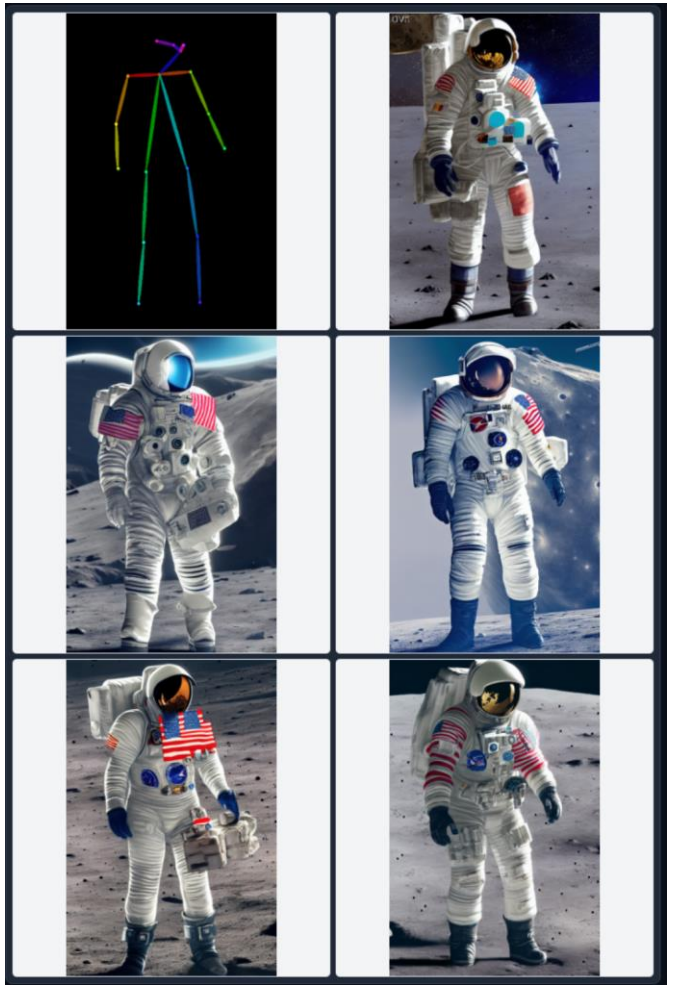


Literary Works

Image Generation

Midjourney

Danqing

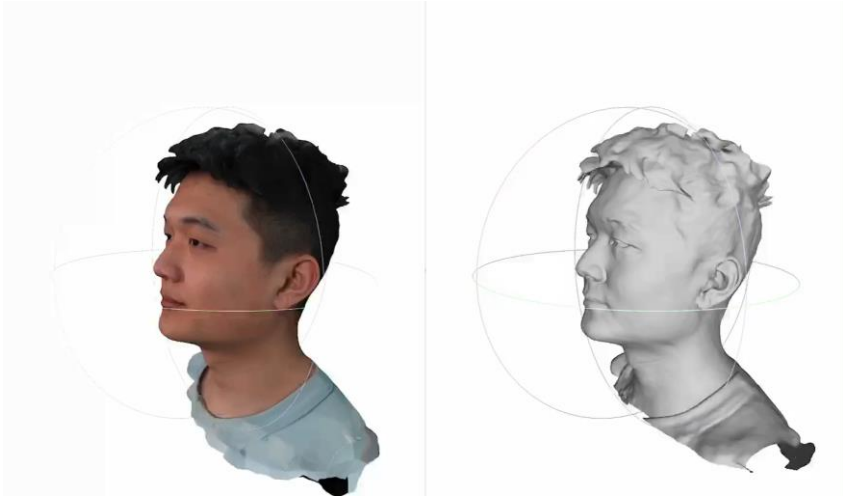


ControlNet

3D & Multi-Modal Generation

GEN-2

Video

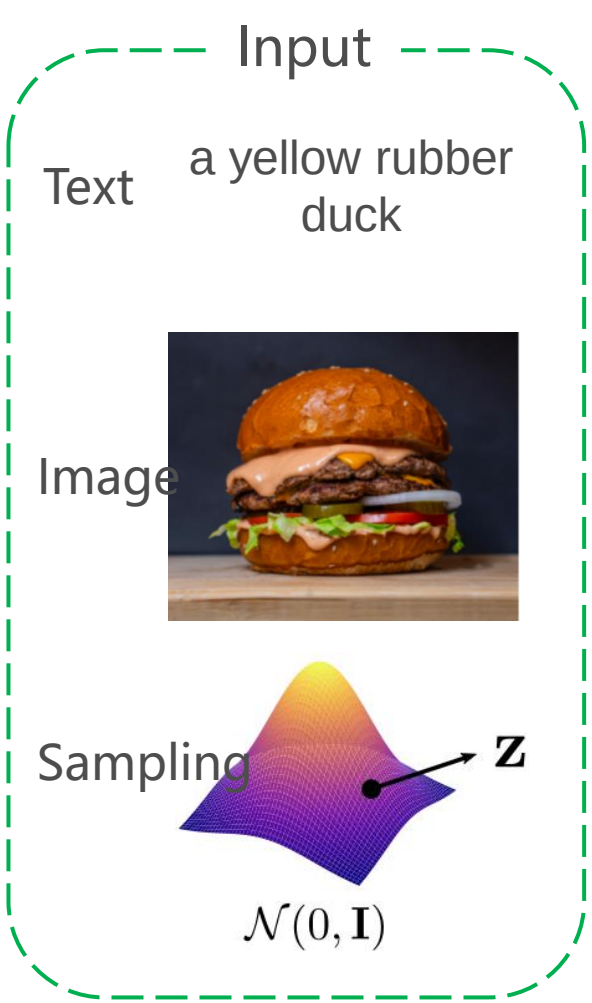


Avatar

3D Object

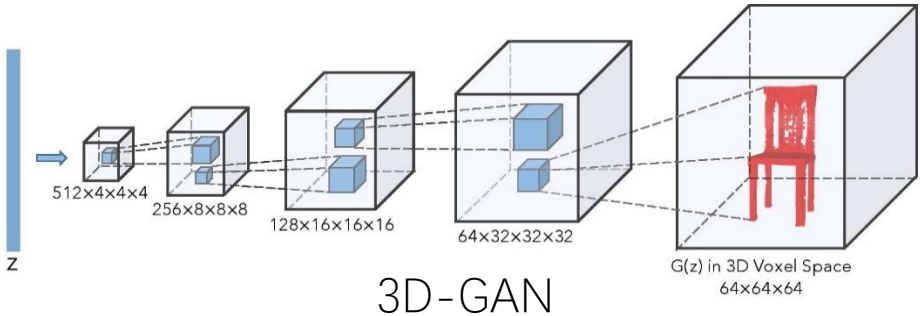
3D Scene

3D AIGC

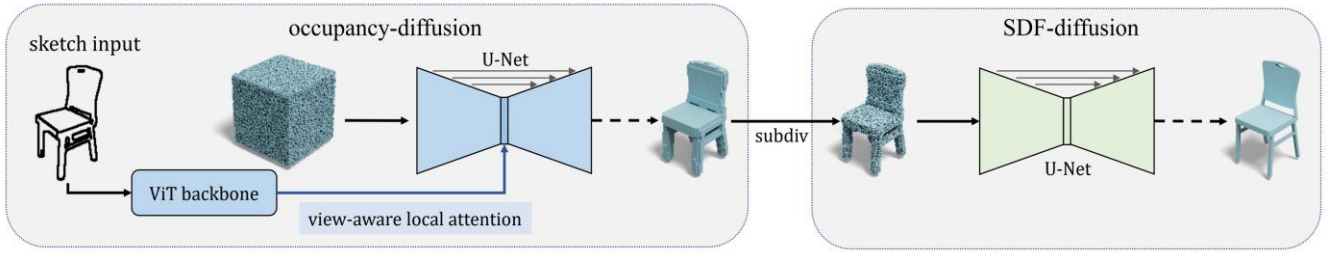

Input
Text
a yellow rubber duck
Image
Sampling
$\mathbf{z}$
$\mathcal{N}(0, \mathbf{I})$
Training data
3D Model
Multi-view
2D Image
Output
Point Cloud
Mesh
NeRF

Status

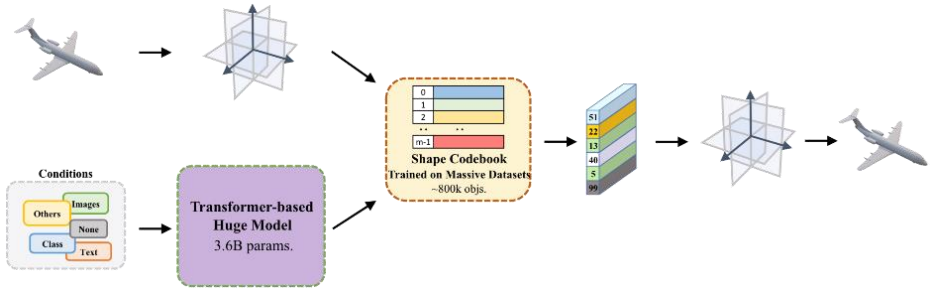
- 3D generation from 3D data
  - **GAN based**: 3D-GAN, Get3D
  - **Diffusion + Volume**: Rodin, LAS-diffusion…
  - **Autoregression**: Argus-3D, PolyGen
  - **LLM + procedural modeling**: 3D-GPT



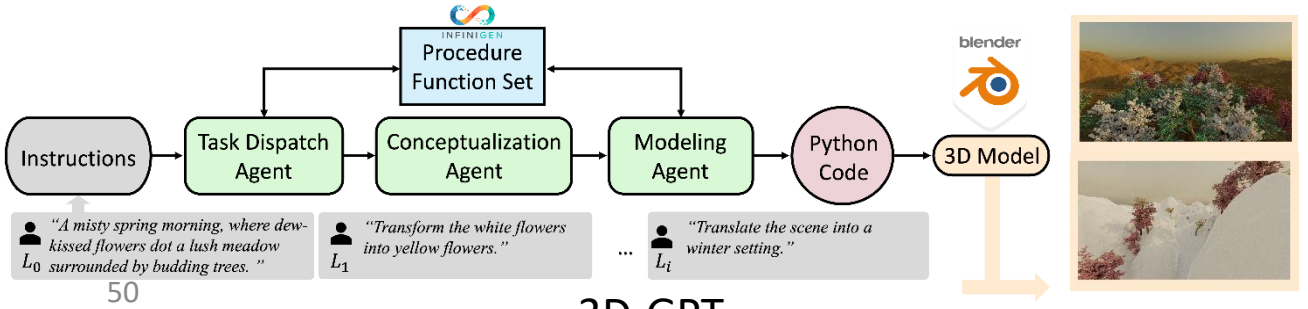
3D-GAN



LAS-diffusion



Argus-3D



3D-GPT

Status

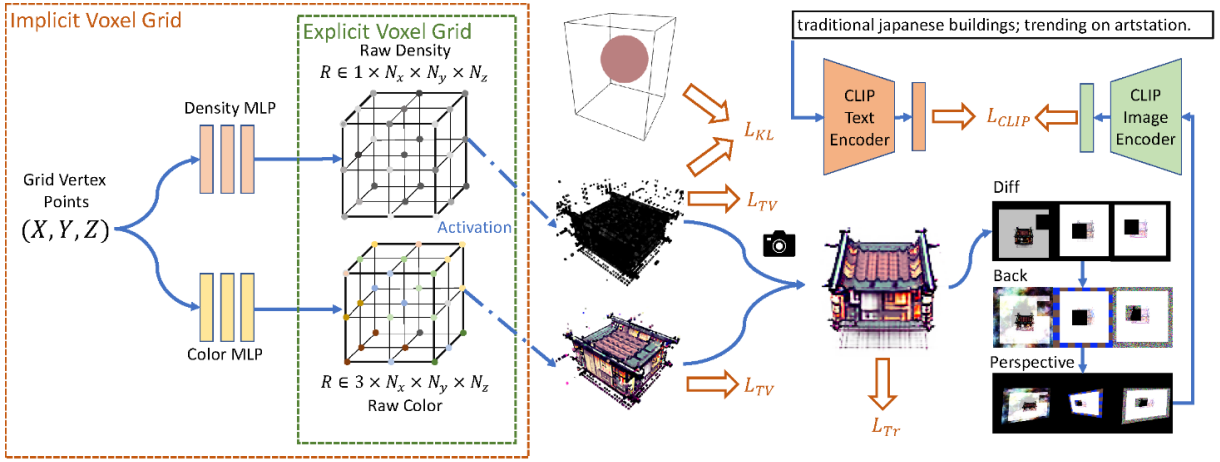
- 3D generation from 3D data

LAION-5B: 5.85 Billion of Images with Text

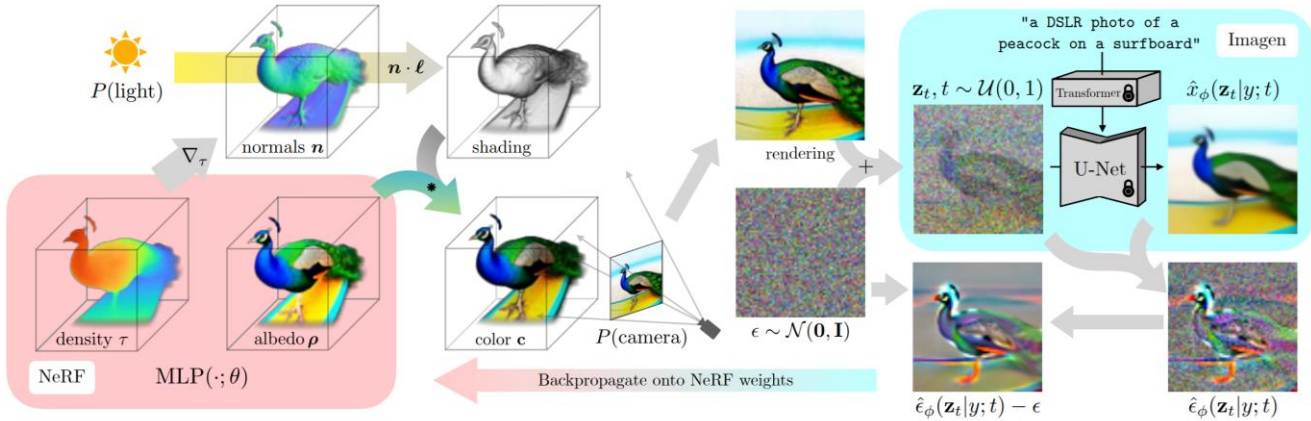
Objaverse-XL: 10 Million of 3D Objects

Status

- 3D Generation from 2D Data
  - **CLIP-guided 3D optimization**: PureCLIPNeRF, CLIP-mesh, Dream3D…
  - **Diffusion-based 3D optimization**: DreamFusion, Magic3D, Fantasia3D, ProlificDreamer…



PureCLIPNeRF



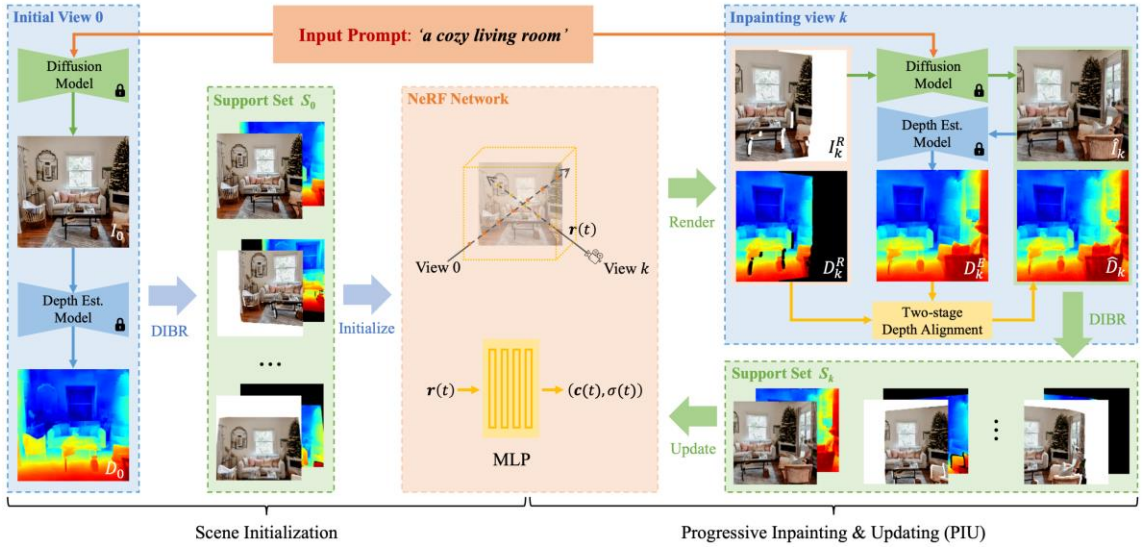
DreamFusion

Status

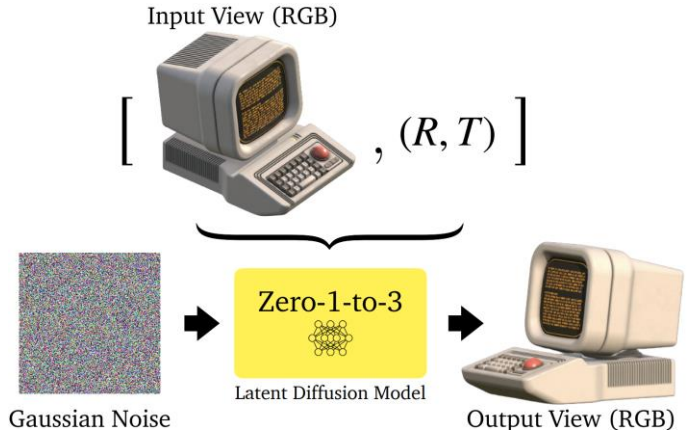
- 3D Generation from 2D Data
  - **CLIP-guided 3D optimization**: PureCLIPNeRF, CLIP-mesh, Dream3D…
  - **Diffusion-based 3D optimization**: DreamFusion, Magic3D, Fantasia3D, ProlificDreamer…

Status

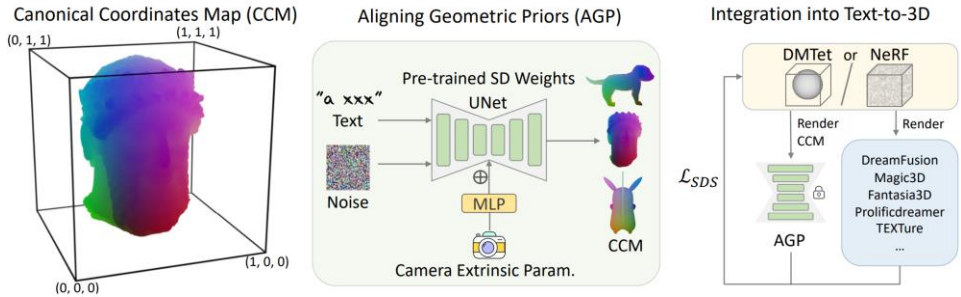
- 3D generation from hybrid data
  - **Progressive Generation**: Text2NeRF, MVDiffusion…
  - **Viewpoint-conditioned Diffusion**: Zero-1-2-3, MVDreamer, SyncDreamer, Wonder3D…
  - **3D-enhanced 2D Diffusion**: SweetDreamer



Text2NeRF



Zero-1-2-3



SweetDreamer