Detection, Segmentation, Visualization and Understanding

Basics and Applications of Deep Learning

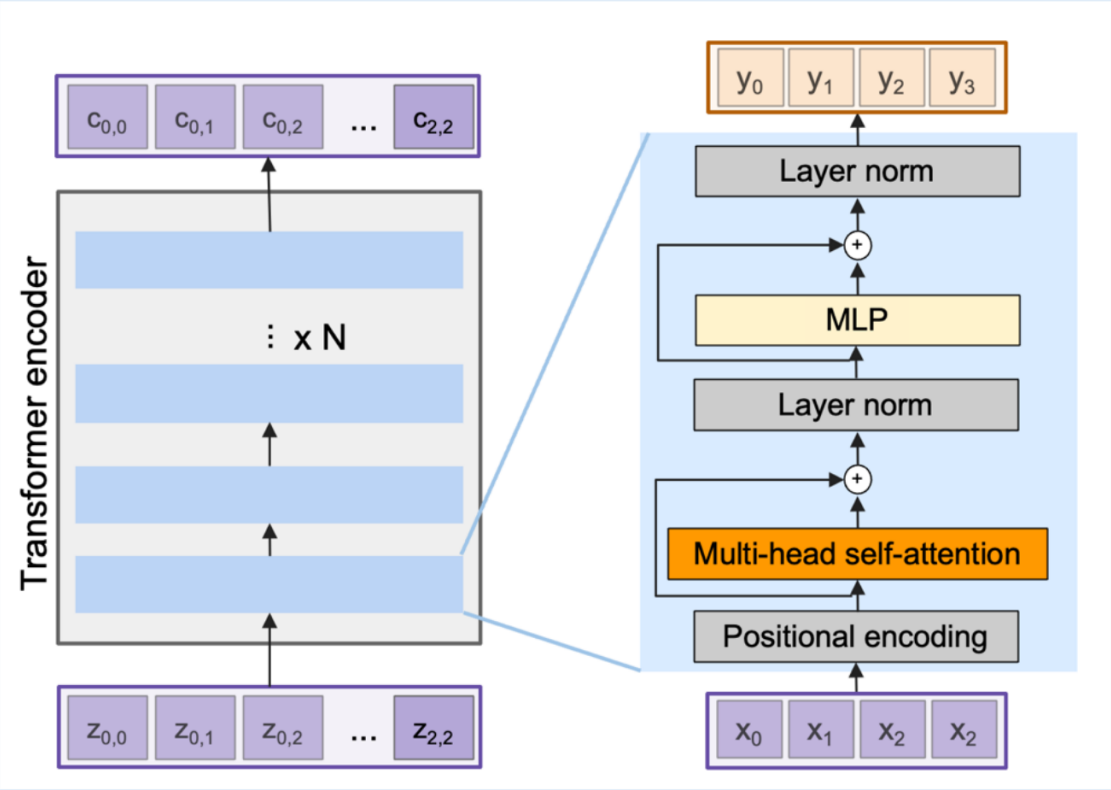
Fall 2026

Department of Mathematical Data Science

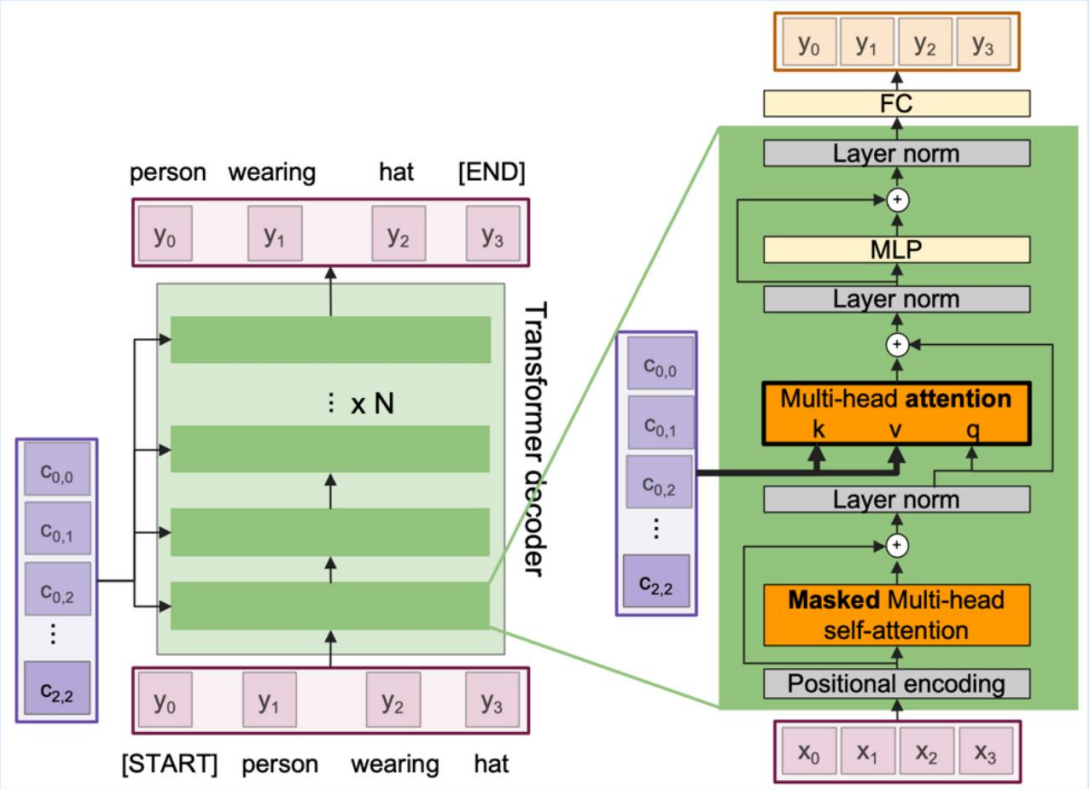
Hanyang University ERICA

Lecturer: Sehyun Kwon

Last time: Transformer



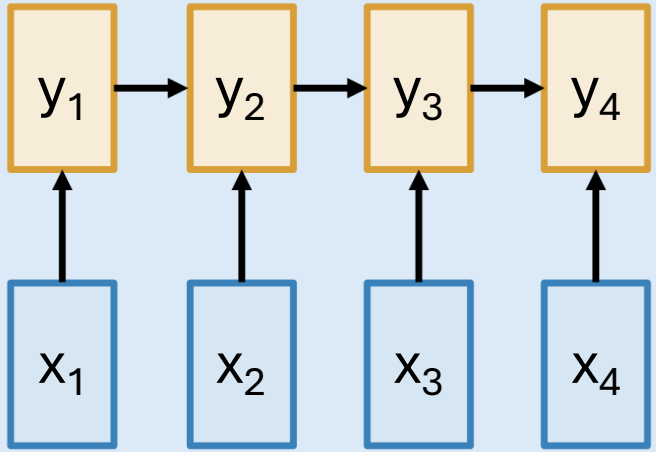
Encoder



Decoder

Three Ways of Processing Sequences

Recurrent Neural Network

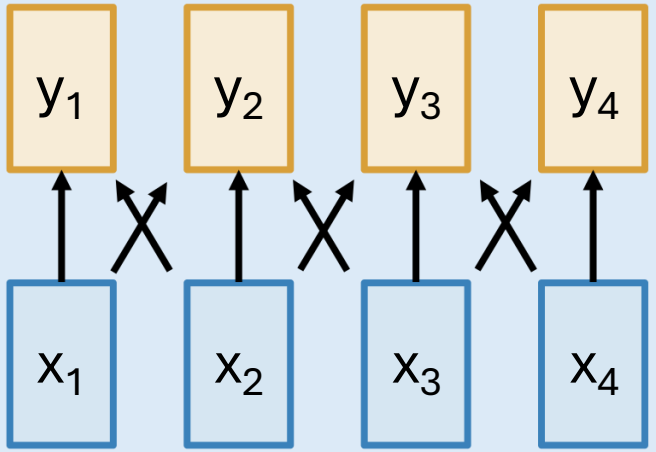


Works on 1D ordered sequences

(+) Theoretically good at long sequences: O(N) compute and memory for a sequence of length N

(-) Not parallelizable. Need to compute hidden states sequentially

Convolution

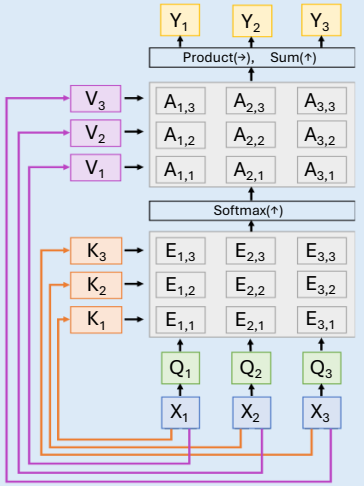


Works on N-dimensional grids

(-) Bad for long sequences: need to stack many layers to build up large receptive fields

(+) Parallelizable, outputs can be computed in parallel

Self-Attention



Works on sets of vectors

(+) Great for long sequences; each output depends directly on all inputs

(+) Highly parallel, it's just 4 matmuls

(−) Expensive: $O(N^2)$ compute, O(N) memory for sequence of length N

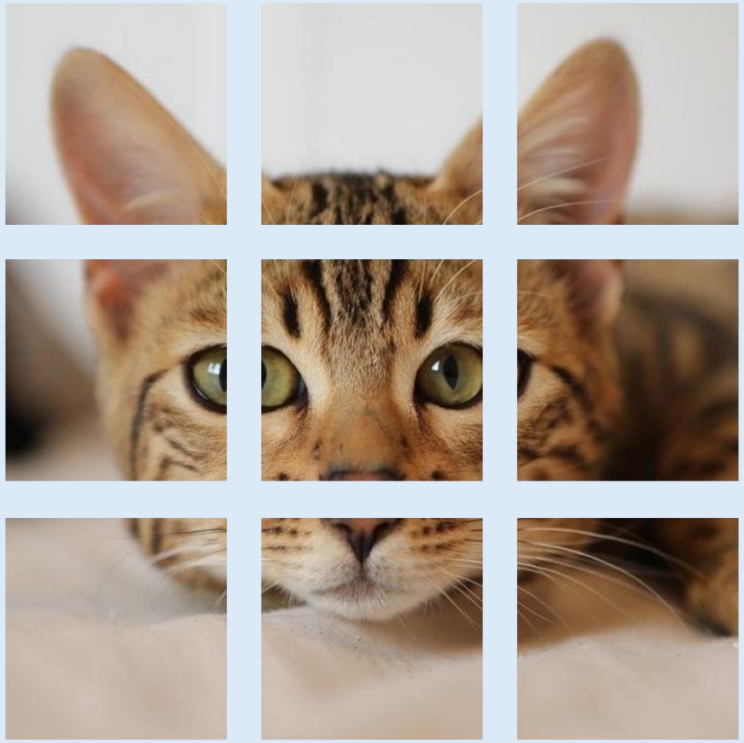
Vision Transformers (ViT)



Input image:
e.g. 224x224x3

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Vision Transformers (ViT)



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformers (ViT)

N input patches, each of shape 3x16x16



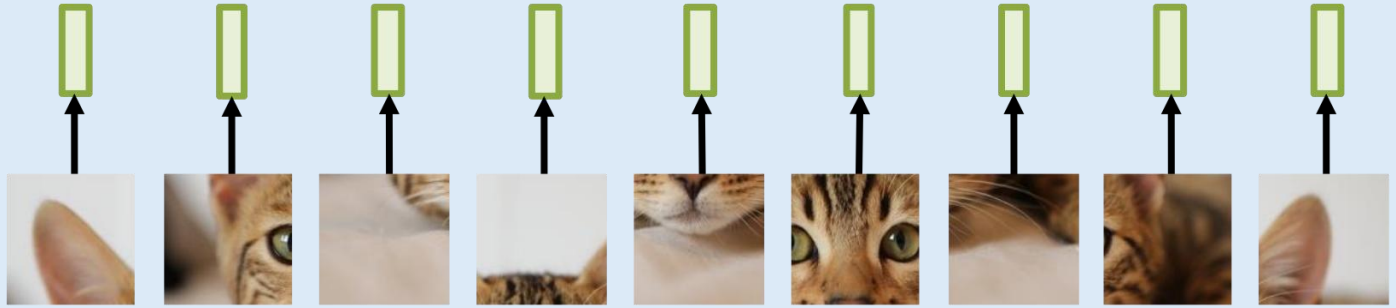
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformers (ViT)

Linear projection to D-dimensional vector

N input patches, each of shape 3x16x16



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

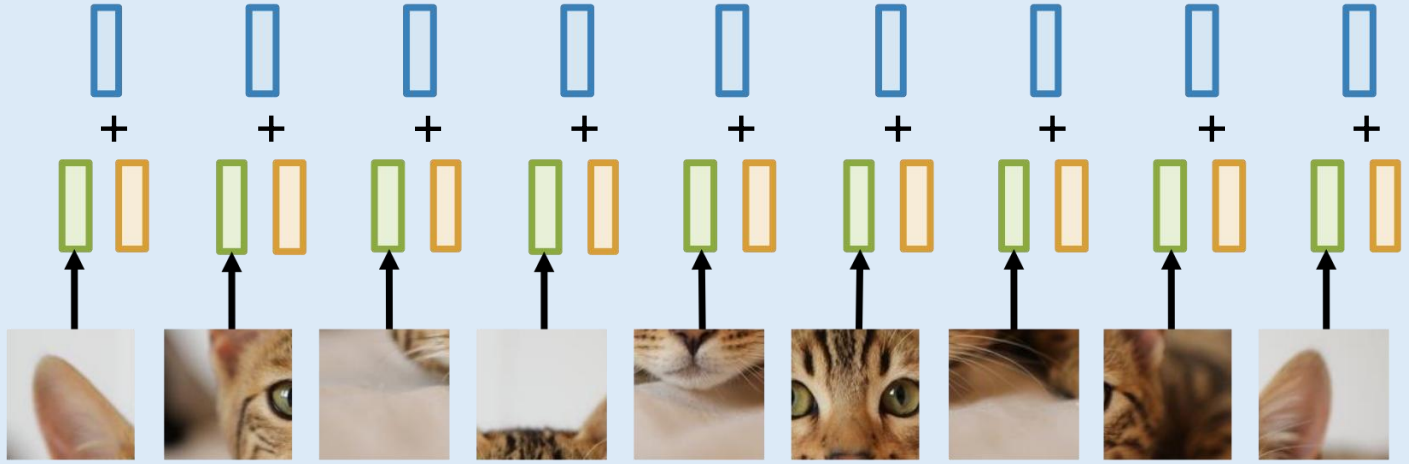
Cat image is free for commercial use under a Pixabay license

Vision Transformers (ViT)

Add positional embedding: learned D-dim vector per position

Linear projection to D-dimensional vector

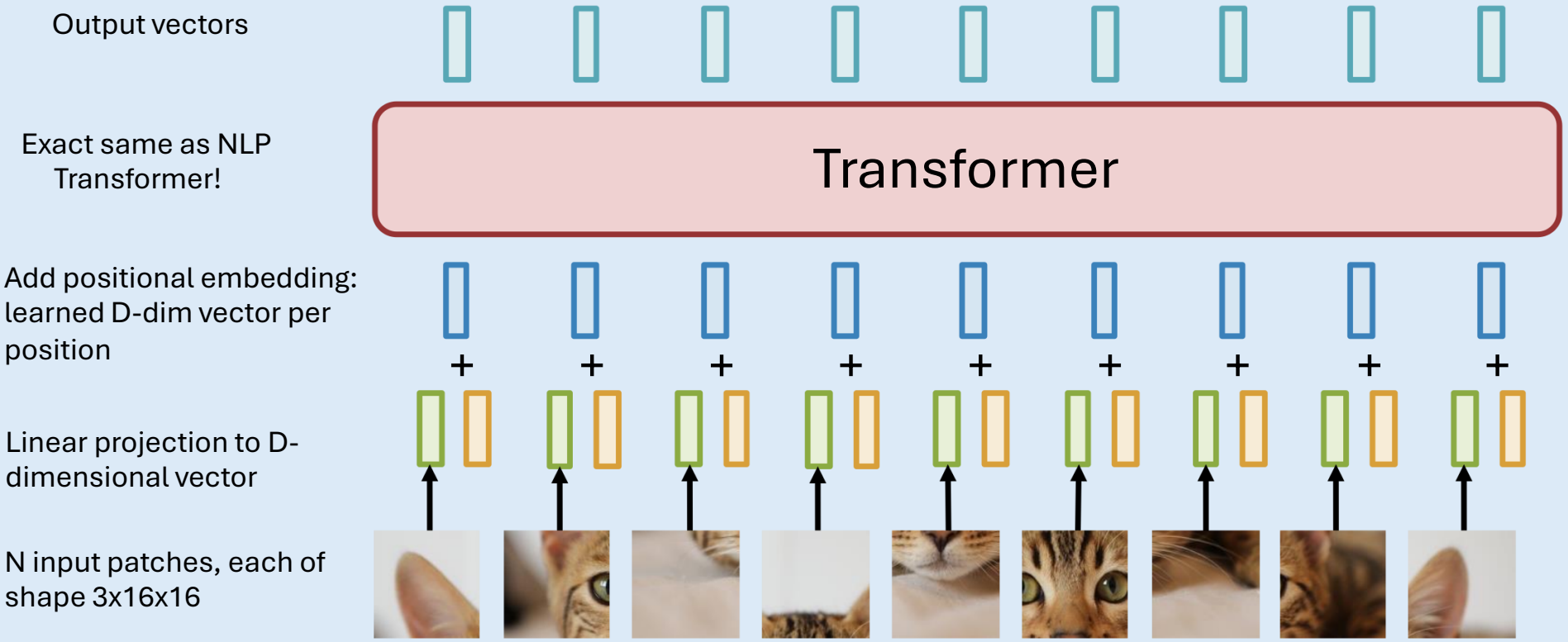
N input patches, each of shape 3x16x16



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

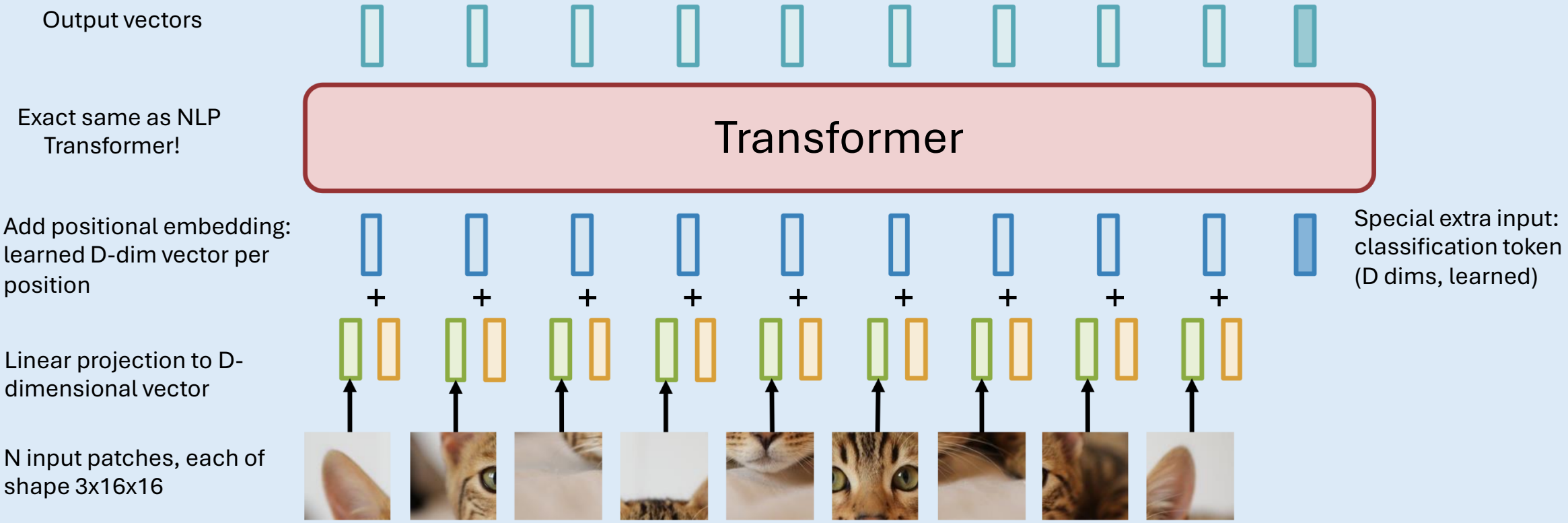
Vision Transformers (ViT)



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

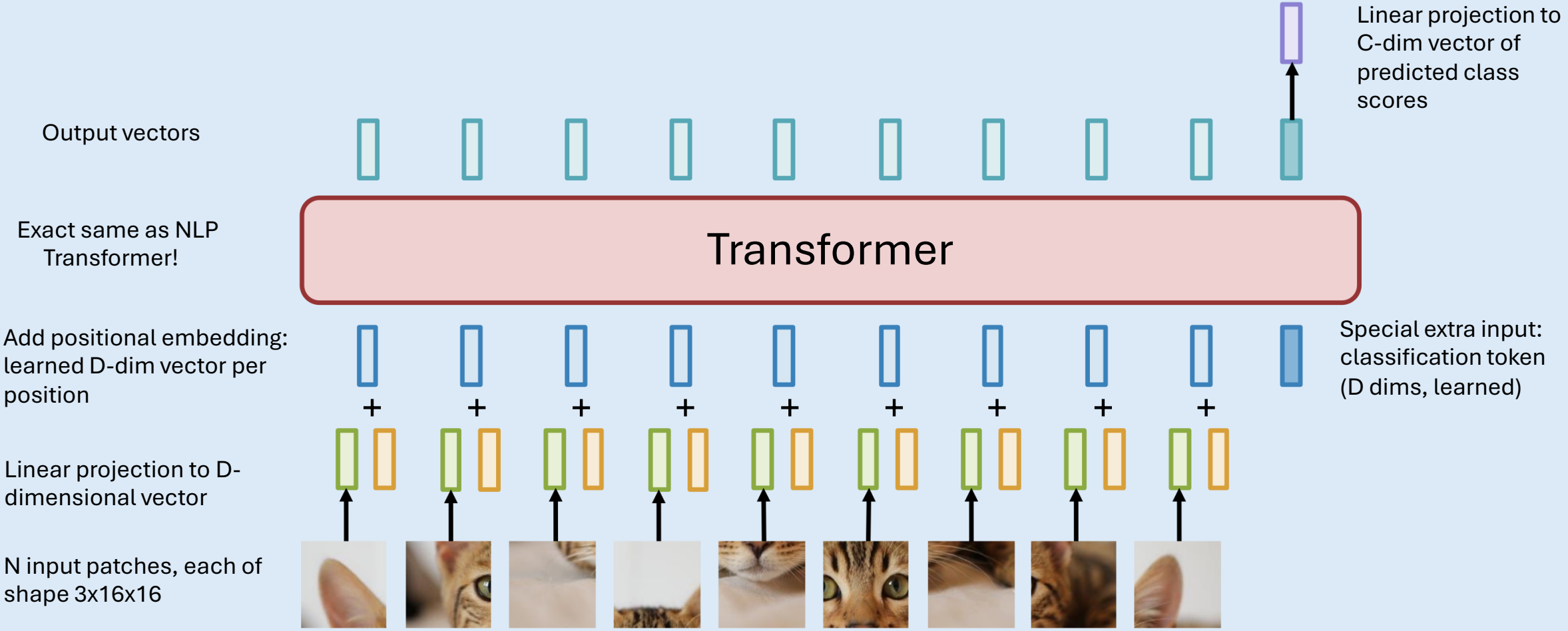
Vision Transformers (ViT)



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

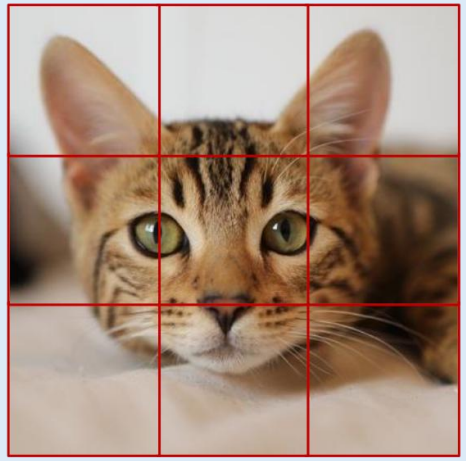
Vision Transformers (ViT)



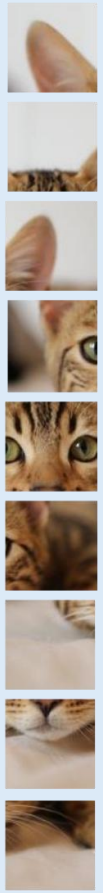
Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Cat image is free for commercial use under a Pixabay license

Vision Transformers (ViT) – a similar approach (different classifier)



Input image:
e.g. 224x224x3



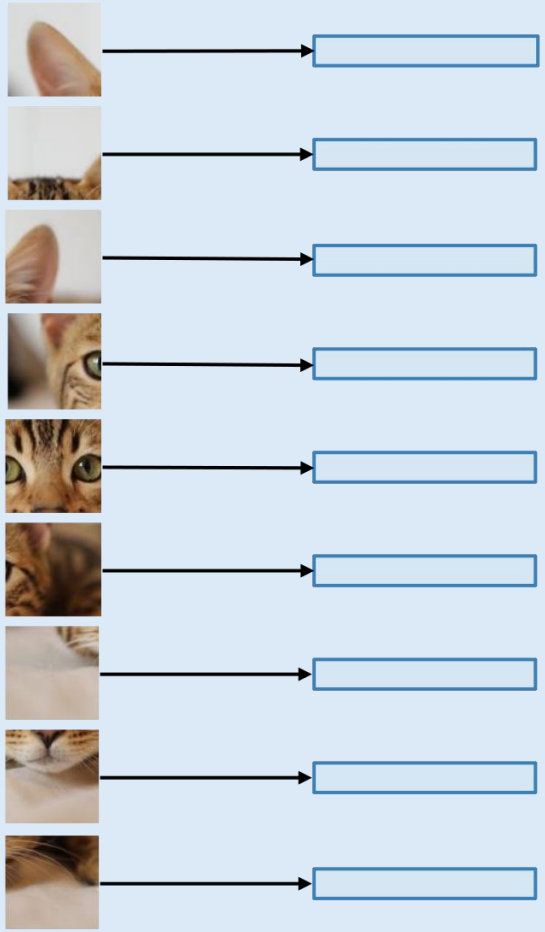
Break into patches
e.g. 16x16x3

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Vision Transformers (ViT)



Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

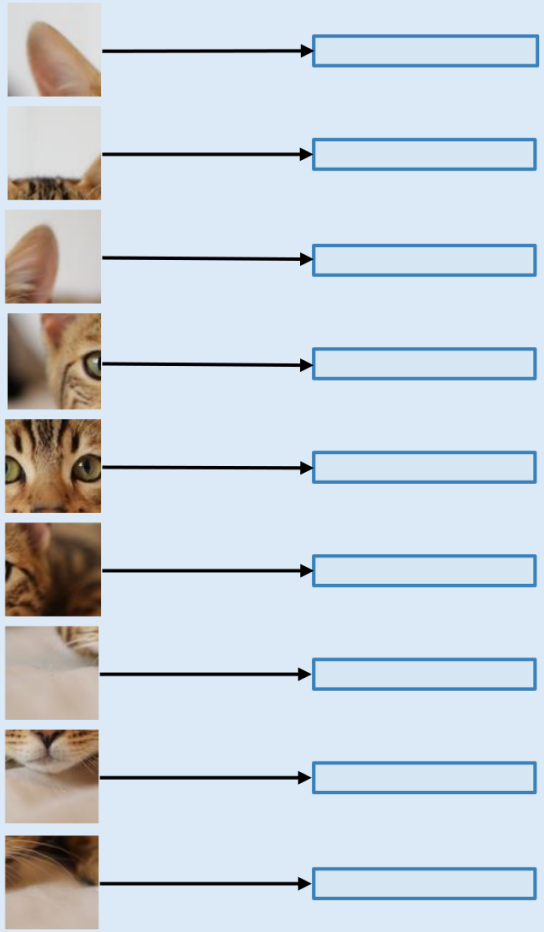
Flatten and apply a linear transform 768 => D

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Vision Transformers (ViT)

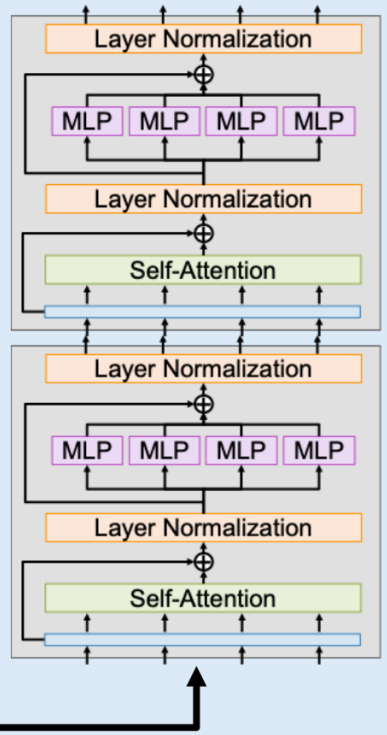


Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D



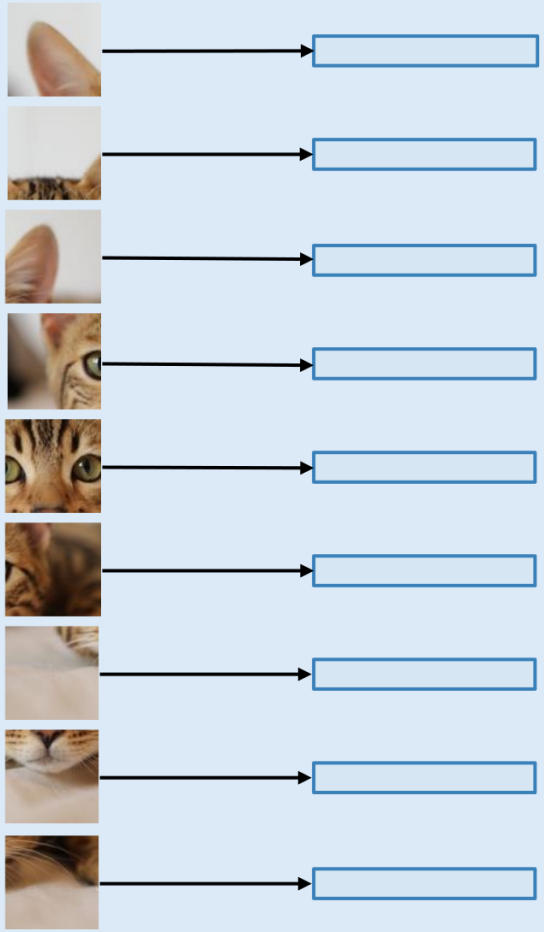
D-dim vector per patch are the input vectors to the Transformer

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Vision Transformers (ViT)

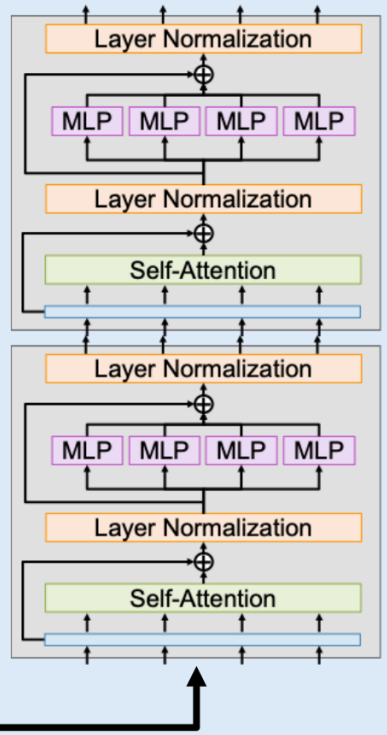


Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D



D-dim vector per patch are the input vectors to the Transformer

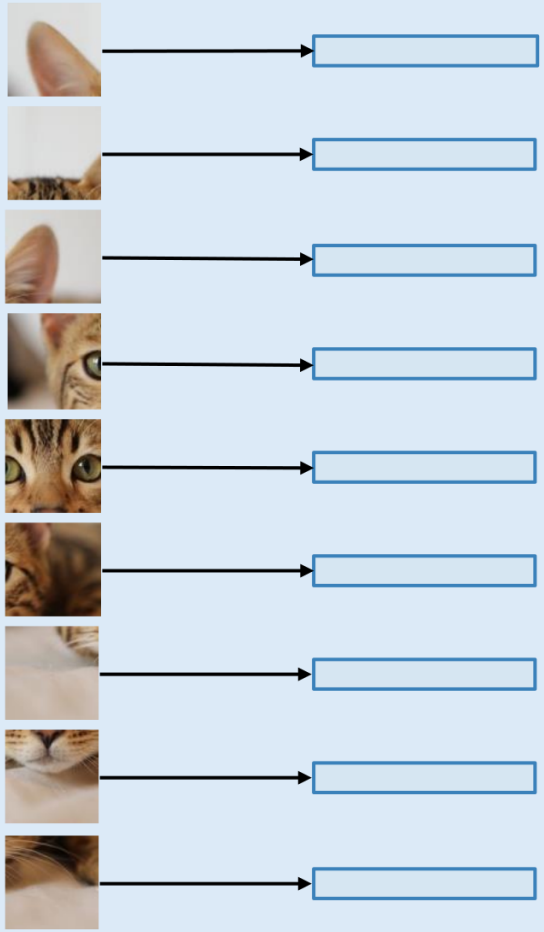
Use positional encoding to tell the transformer the 2D position of each patch

Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Vision Transformers (ViT)

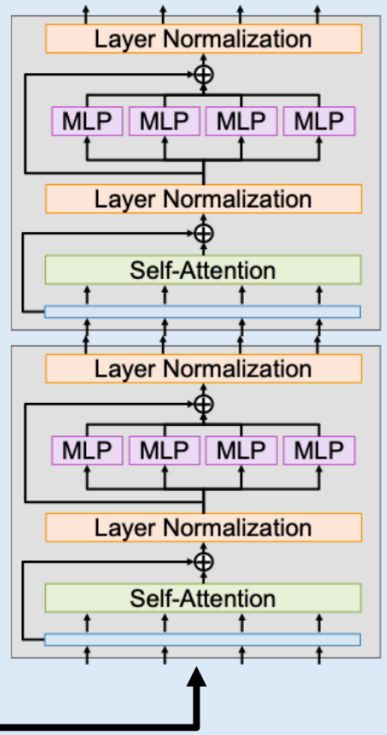


Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D



D-dim vector per patch are the input vectors to the Transformer

Don't use any masking; each image patch can look at all other image patches

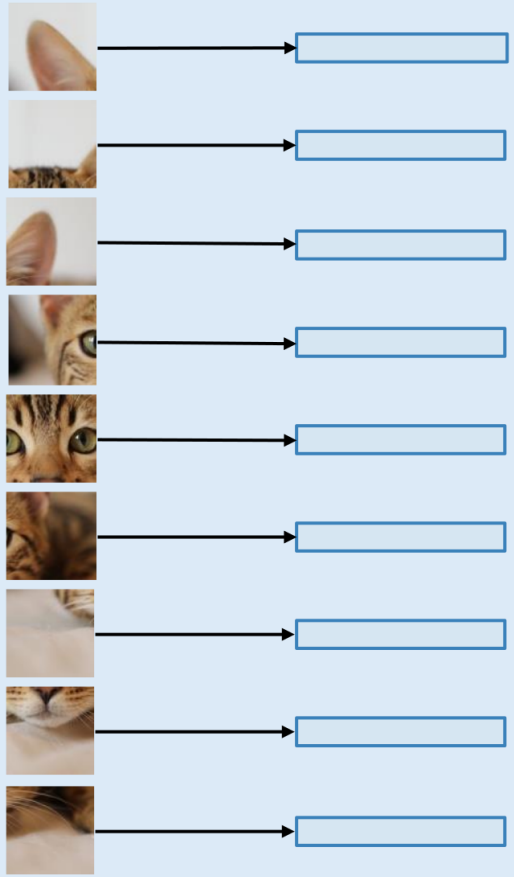
Use positional encoding to tell the transformer the 2D position of each patch

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT)

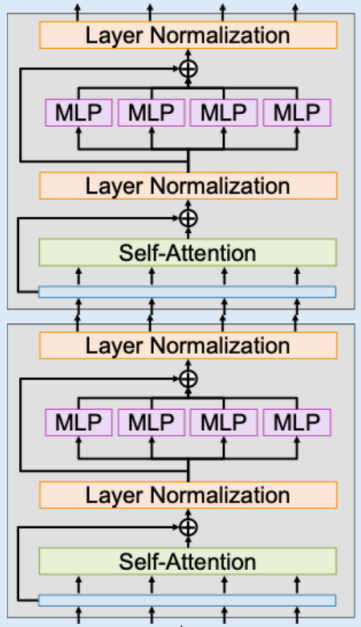


Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D



D-dim vector per patch are the input vectors to the Transformer

Transformer gives an output vector per patch

Don't use any masking; each image patch can look at all other image patches

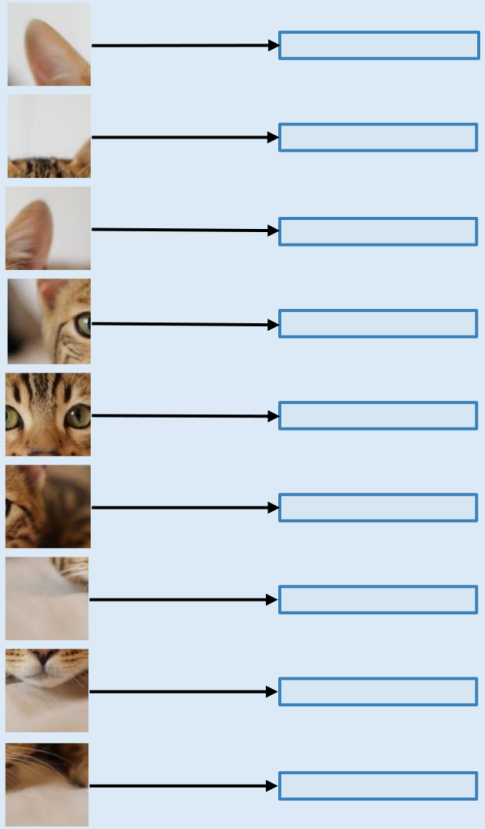
Use positional encoding to tell the transformer the 2D position of each patch

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Vision Transformers (ViT)

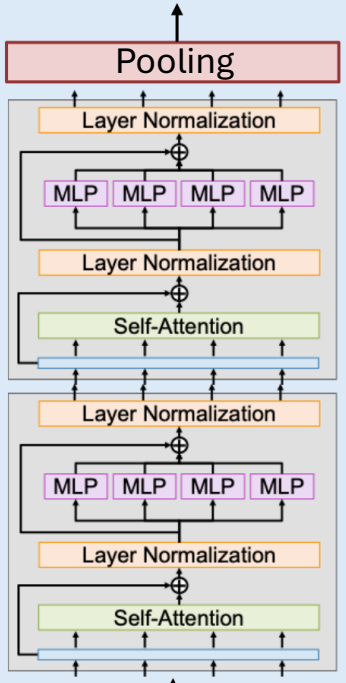


Input image:
e.g. 224x224x3



Break into patches
e.g. 16x16x3

Flatten and apply a linear transform 768 => D



Average pool NxD vectors to 1xD, apply a linear layer D=>C to predict class scores

D-dim vector per patch are the input vectors to the Transformer

Transformer gives an output vector per patch

Don't use any masking; each image patch can look at all other image patches

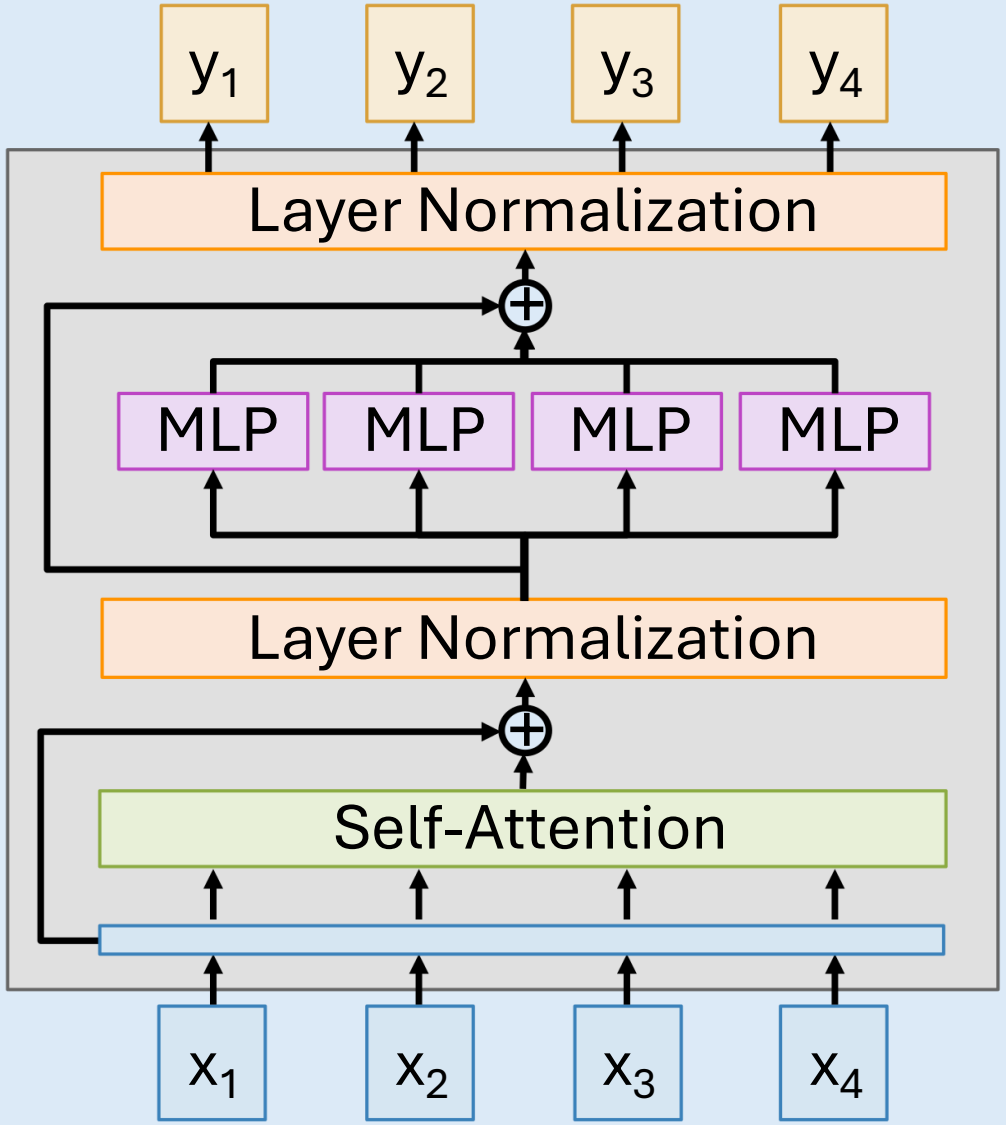
Use positional encoding to tell the transformer the 2D position of each patch

Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ICLR 2021

Tweaking Transformers

The Transformer architecture has not changed much since 2017.

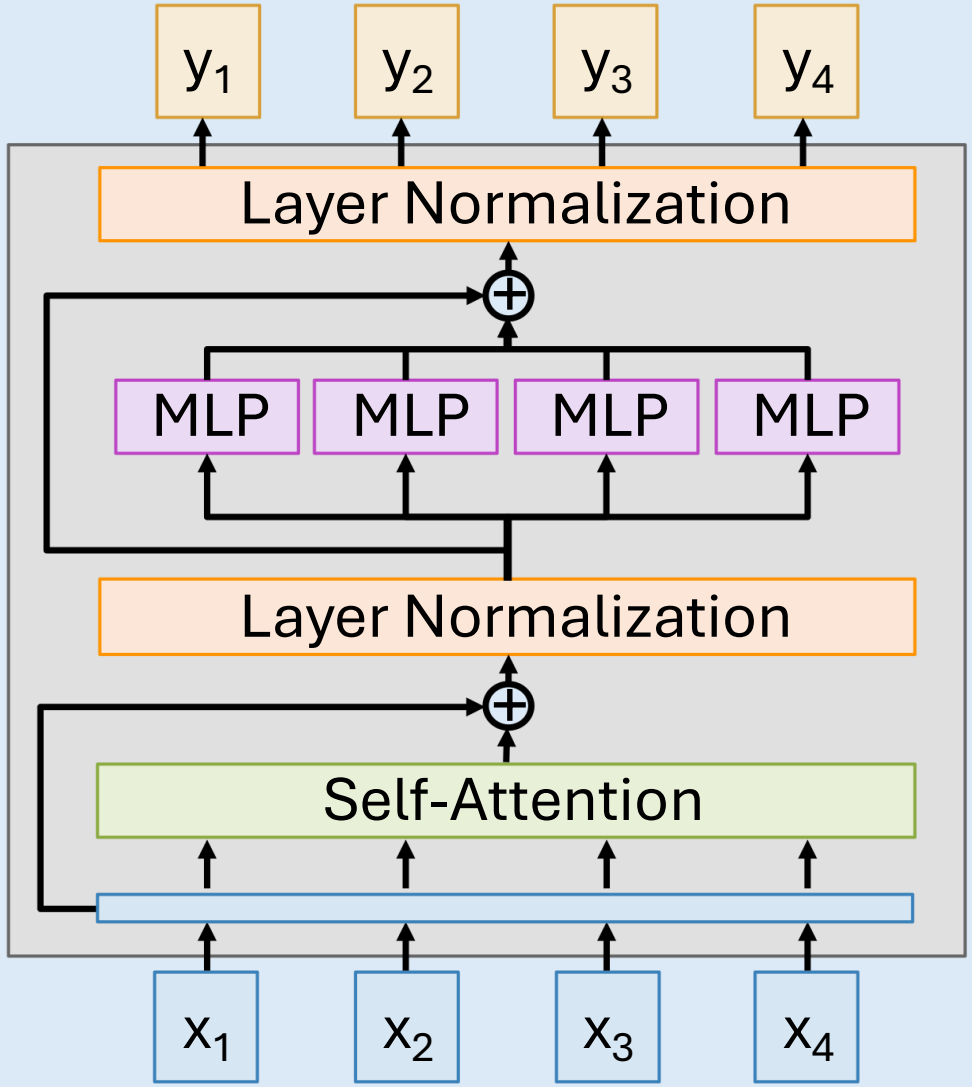
But a few changes have become common:



Pre-Norm Transformer

Layer normalization is outside the residual connections

Kind of weird, the model can't actually learn the identify function



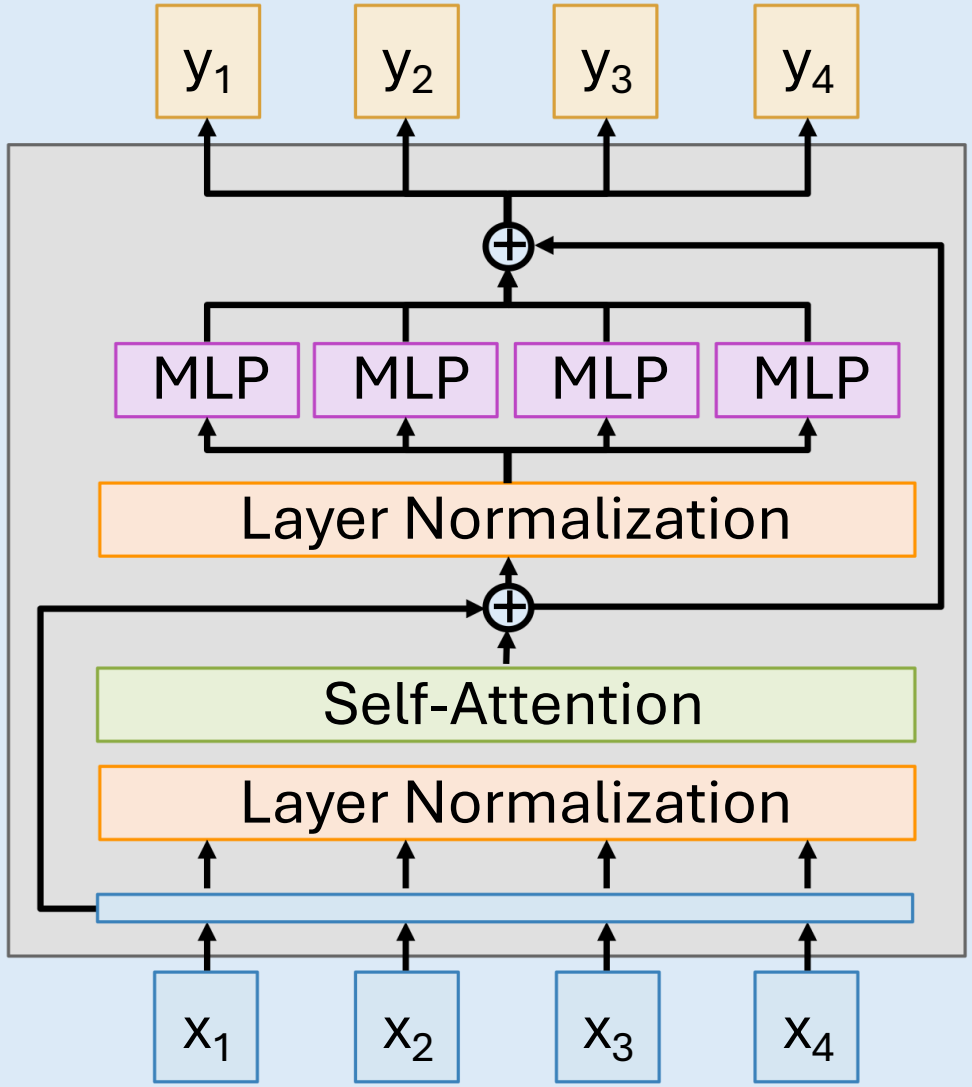
Baevski & Auli, "Adaptive Input Representations for Neural Language Modeling", arXiv 2018

Pre-Norm Transformer

Layer normalization is outside the residual connections

Kind of weird, the model can't actually learn the identify function

Solution: Move layer normalization before the Self-Attention and MLP, inside the residual connections. Training is more stable.



Baevski & Auli, "Adaptive Input Representations for Neural Language Modeling", arXiv 2018

RMSNorm

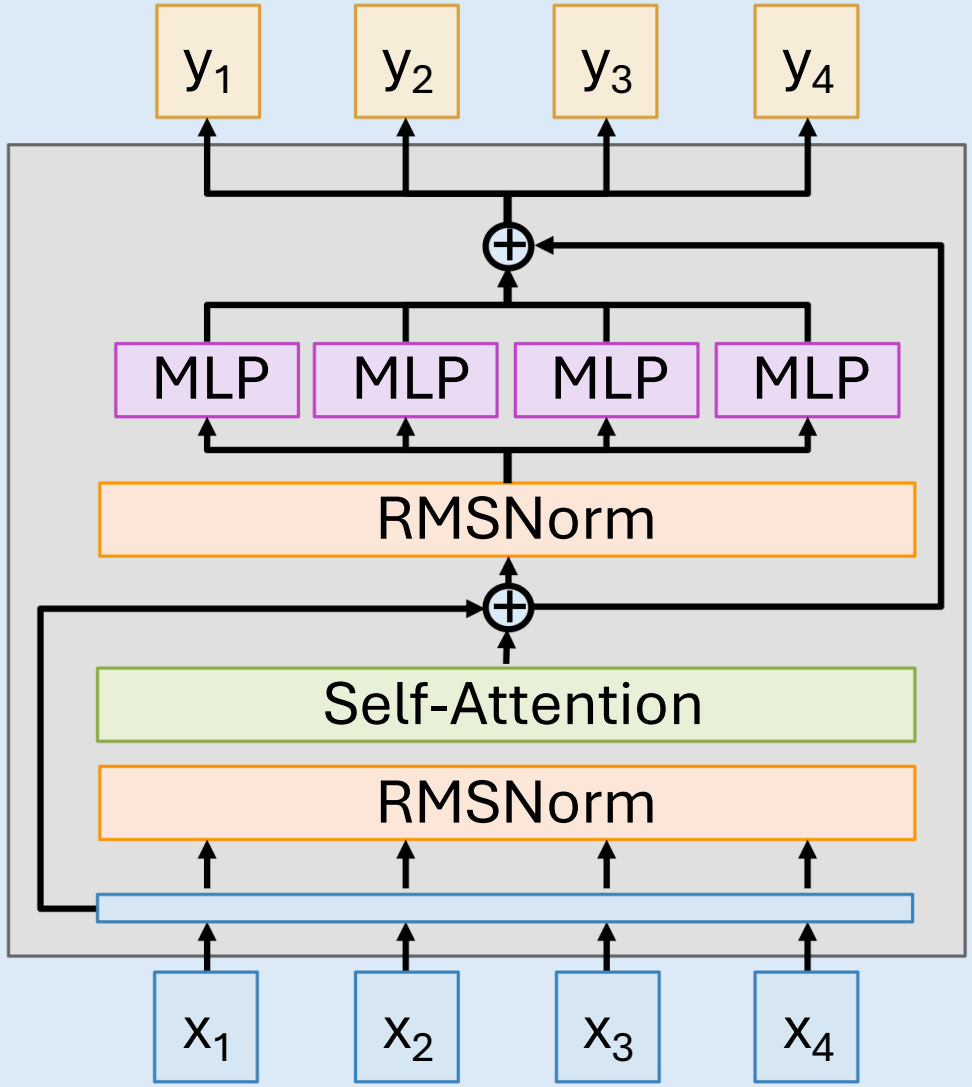
Replace Layer Normalization with Root-Mean-Square Normalization (RMSNorm)

Input: x [shape D]
Output: y [shape D]
Weight: γ [shape D]

$$y_i = \frac{x_i}{\mathrm{RMS}(x)} \cdot \gamma_i$$

$$\mathrm{RMS}(x) = \sqrt{\varepsilon + \frac{1}{N}\sum_{i=1}^{N} {x_i}^2}$$

Training is a bit more stable



Zhang and Sennrich, "Root Mean Square Layer Normalization", NeurIPS 2019

SwiGLU MLP

Classic MLP:

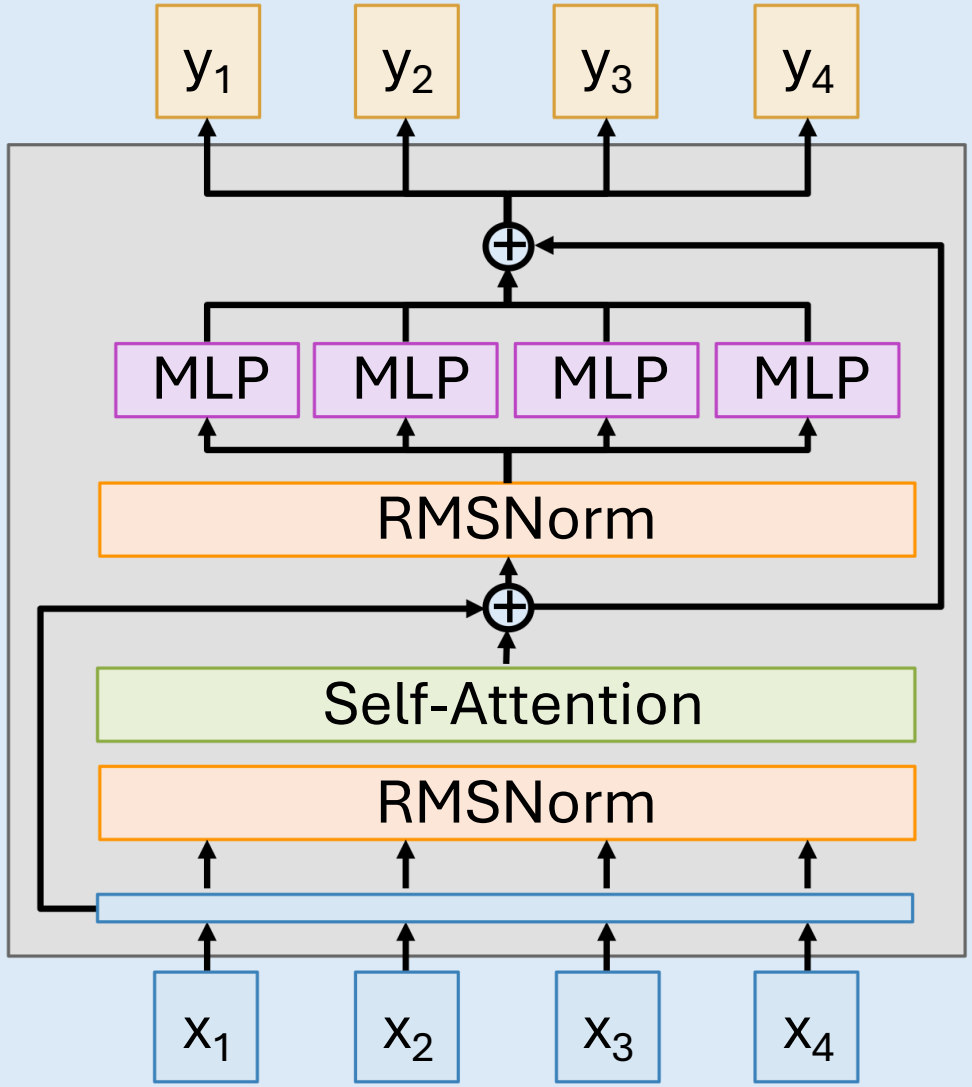
Input: X [N x D]
Weights: W_1 [D x 4D]
W_2 [4D x D]
Output: $Y = \sigma(XW_1)W_2$ [N x D]

SwiGLU MLP:

Input: X [N x D]
Weights: W_1 , W_2 [D x H]
W_3 [H x D]
Output:

$$Y = (\mathrm{SiLU}(XW_1) \odot XW_2)W_3$$

Setting H = 8D/3 keeps same total params



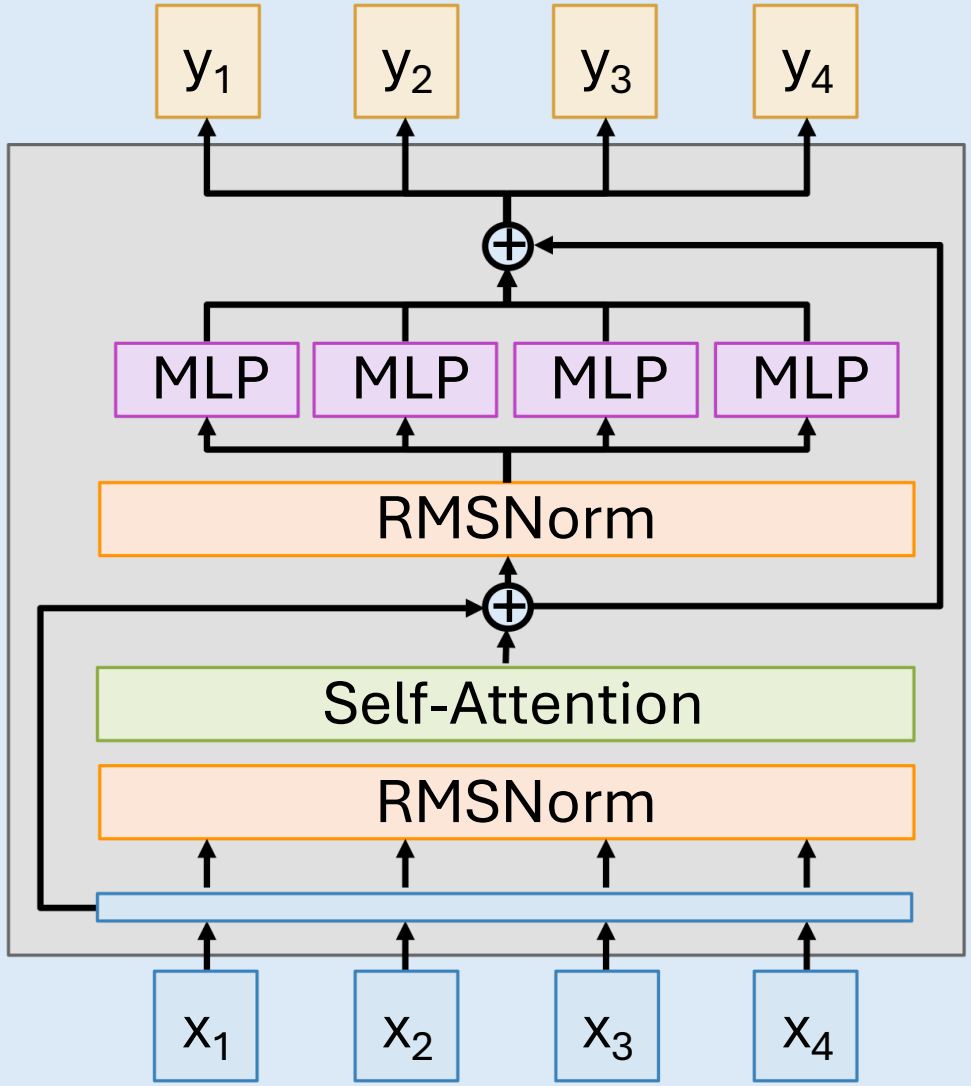
Shazeer, "GLU Variants Improve Transformers", 2020

Mixture of Experts (MoE)

Learn E separate sets of MLP weights in each block; each MLP is an expert

W_1: [D x 4D] => [E x D x 4D]
W_2: [4D x D] => [E x 4D x D]

$$Y = (\mathrm{SiLU}(XW_1) \odot XW_2)W_3$$



Shazeer et al, “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer”, 2017

Mixture of Experts (MoE)

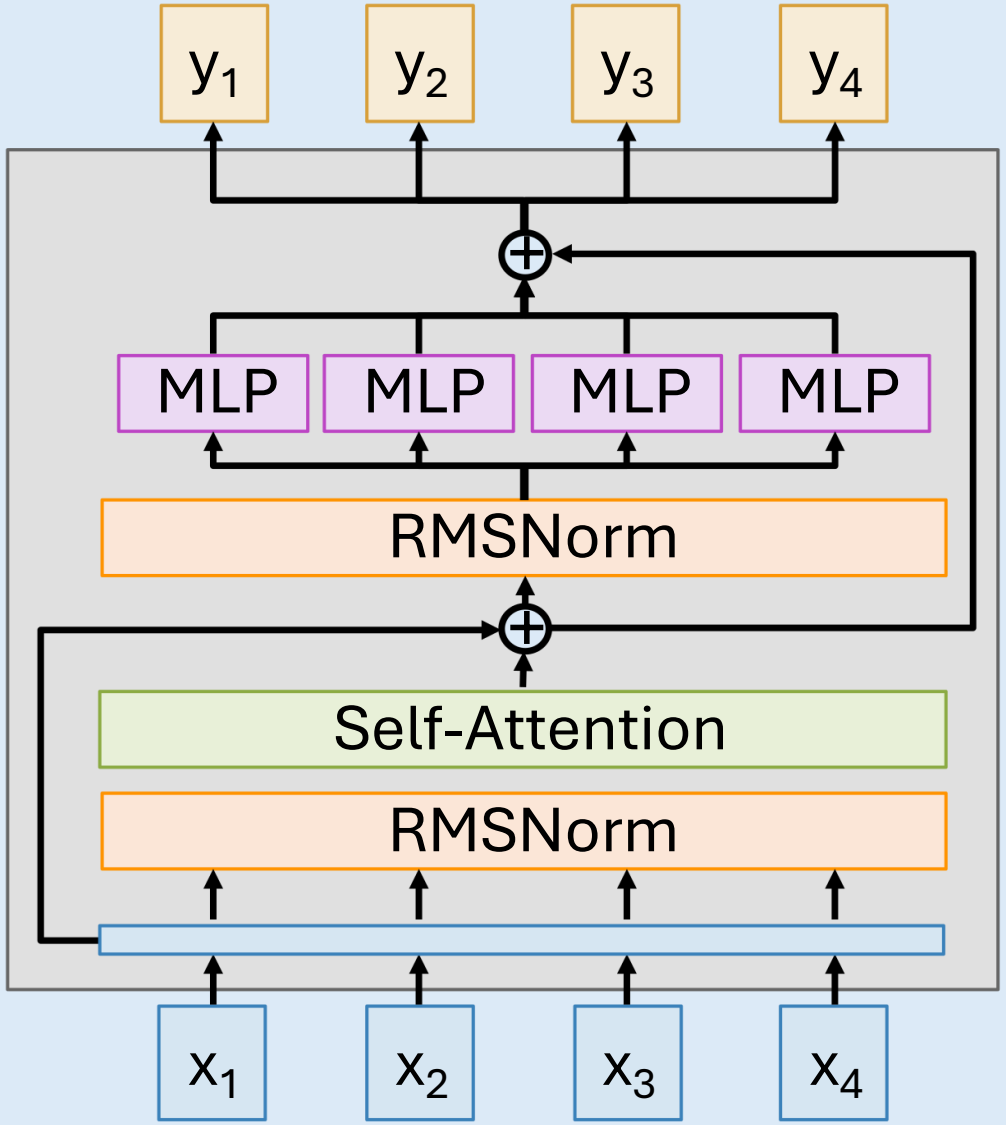
Learn E separate sets of MLP weights in each block; each MLP is an expert

W_1: [D x 4D] => [E x D x 4D]
W_2: [4D x D] => [E x 4D x D]

Each token gets routed to A < E of the experts. These are the active experts.

Increases params by E,
But only increases compute by A

All of the biggest LLMs today (e.g. GPT4o, GPT4.5, Claude 3.7, Gemini 2.5 Pro, etc) almost certainly use MoE and have > 1T params; but they don't publish details anymore

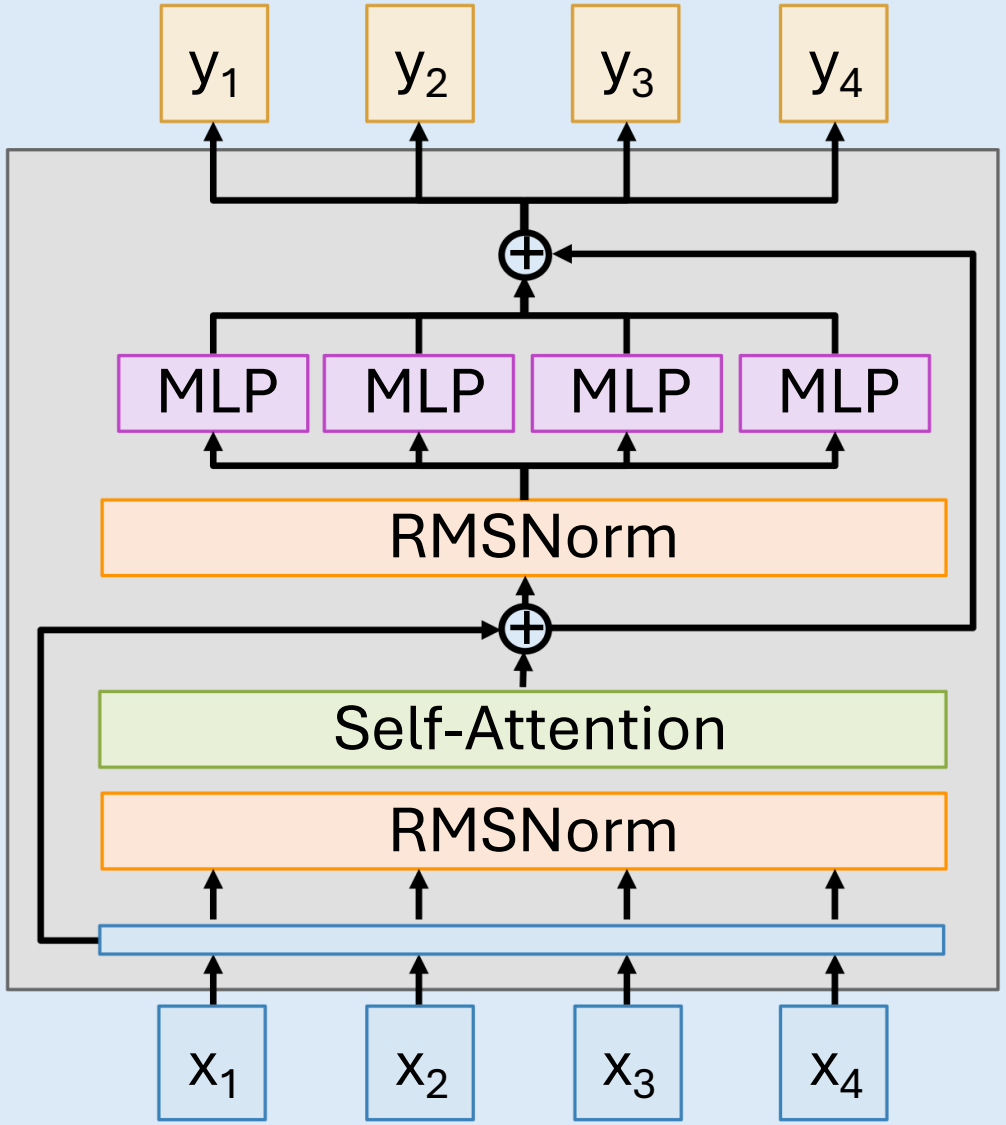


Tweaking Transformers

The Transformer architecture has not changed much since 2017.

But a few changes have become common:

- Pre-Norm: Move normalization inside residual
- RMSNorm: Different normalization layer
- SwiGLU: Different MLP architecture
- Mixture of Experts (MoE): Learn E different MLPs, use A < E of them per token. Massively increase params, modest increase to compute cost.



Today

- Transformers Recap
- **Computer Vision Tasks**
  - Semantic Segmentation
  - Object Detection
  - Instance Segmentation
- Visualization & Understanding
  - Model Layers Visualization
  - Saliency Maps
  - CAM & Grad-CAM

Computer Vision Tasks

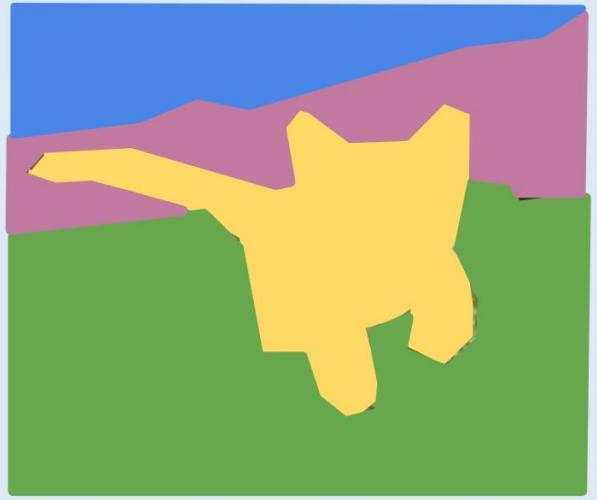
Classification



CAT

No spatial extent

Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

This image is CC0 public domain

Detection, Segmentation and Understanding

Recall: Image Classification: A core task in Computer Vision



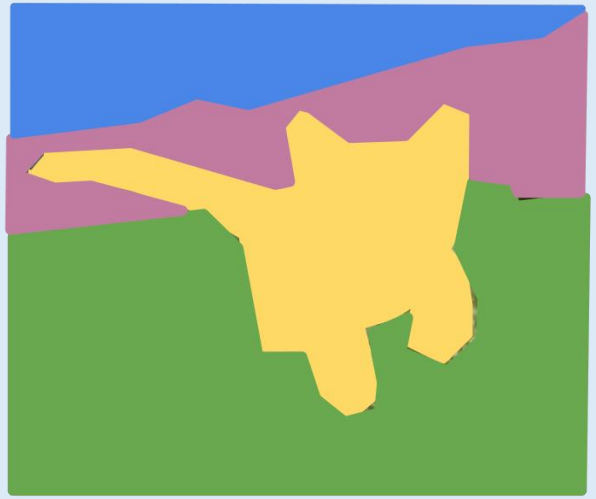
This image by Nikita is licensed under CC-BY 2.0

(assume given a set of possible labels)
{dog, cat, truck, plane, …}

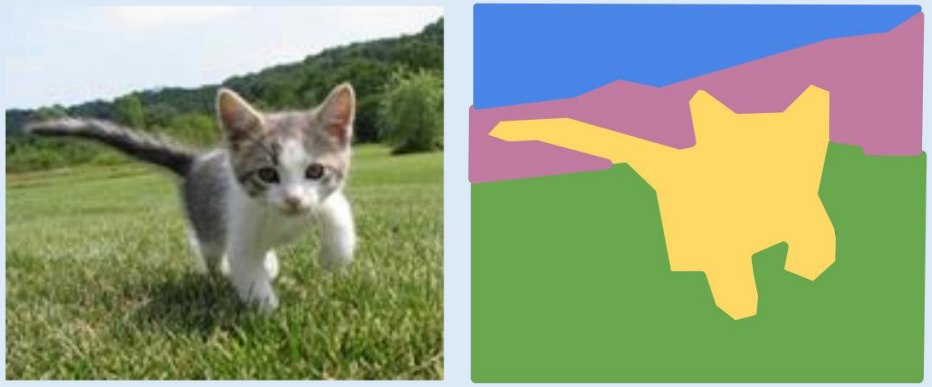


cat

Semantic Segmentation

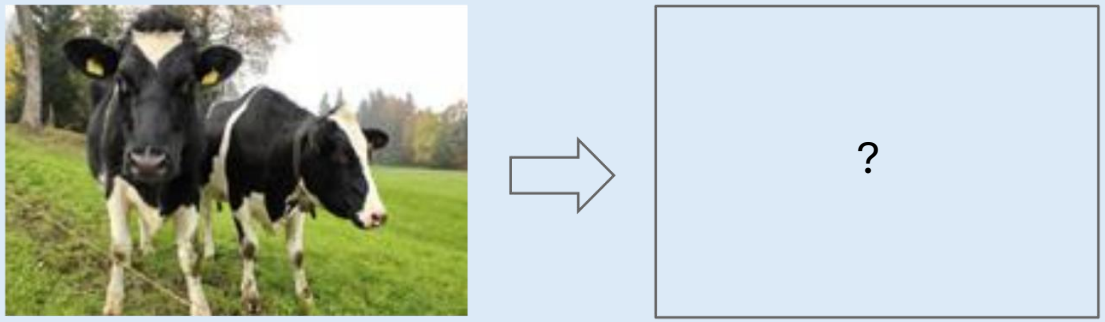
Classification	Semantic Segmentation	Object Detection	Instance Segmentation
			
CAT	GRASS, CAT, TREE, SKY	DOG, DOG, CAT	DOG, DOG, CAT
No spatial extent	No objects, just pixels	Multiple Object	

Semantic Segmentation: The Problem



GRASS, CAT, TREE,
SKY, …

Paired training data: for each training image, each pixel is labeled with a semantic category.



At test time, classify each pixel of a new image.

Semantic Segmentation Idea: Sliding Window

Full image



Semantic Segmentation Idea: Sliding Window

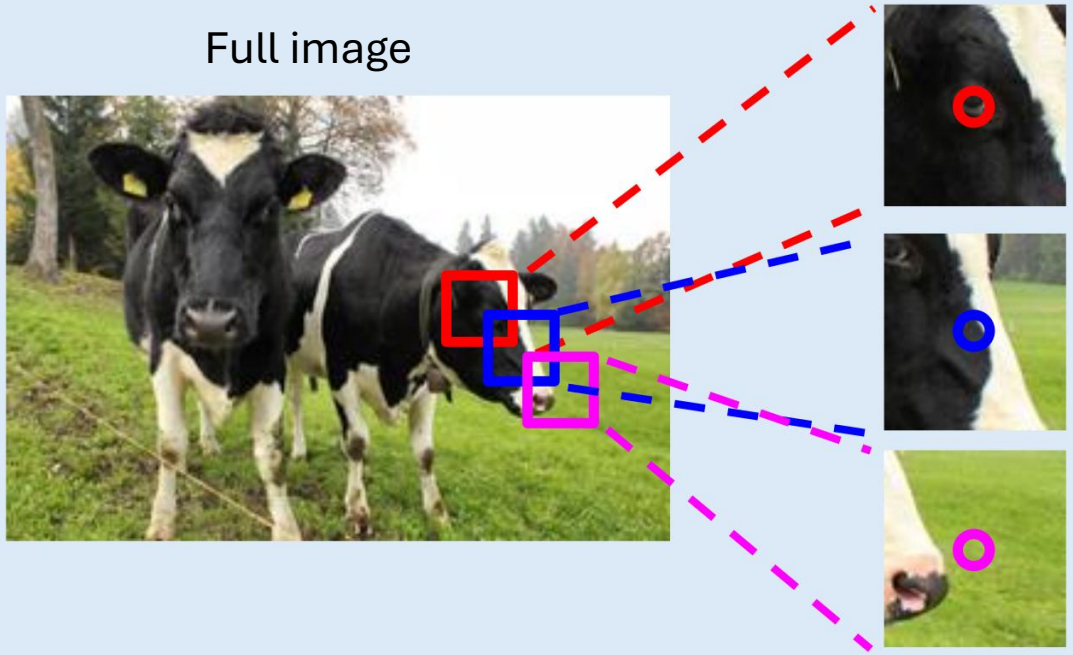
Full image



Impossible to classify without context

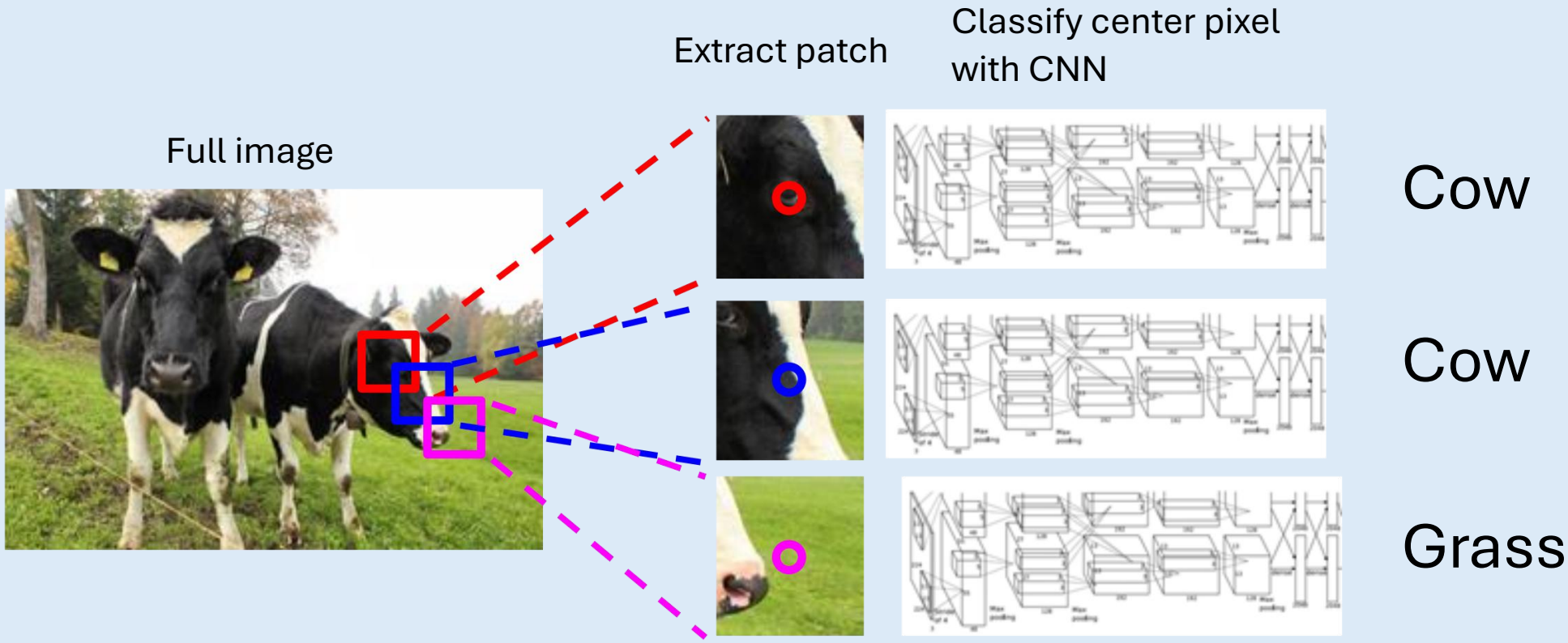
Q: how do we include context?

Semantic Segmentation Idea: Sliding Window



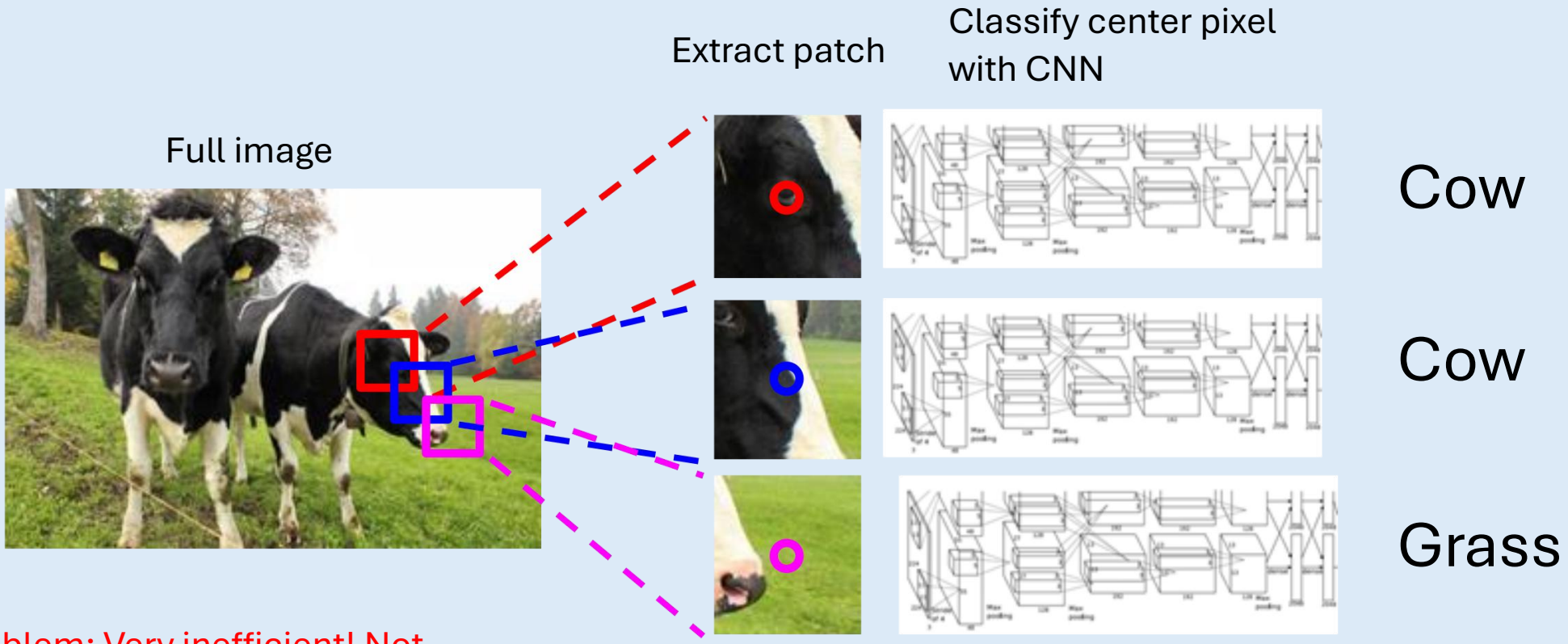
Q: how do we model this?

Semantic Segmentation Idea: Sliding Window



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

Semantic Segmentation Idea: Sliding Window



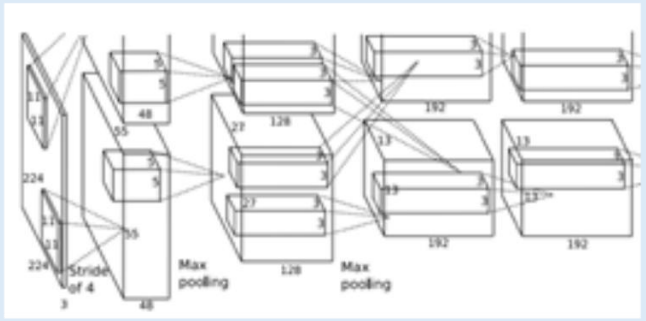
Problem: Very inefficient! Not reusing shared features between overlapping patches

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

Semantic Segmentation Idea: Convolution

Full image





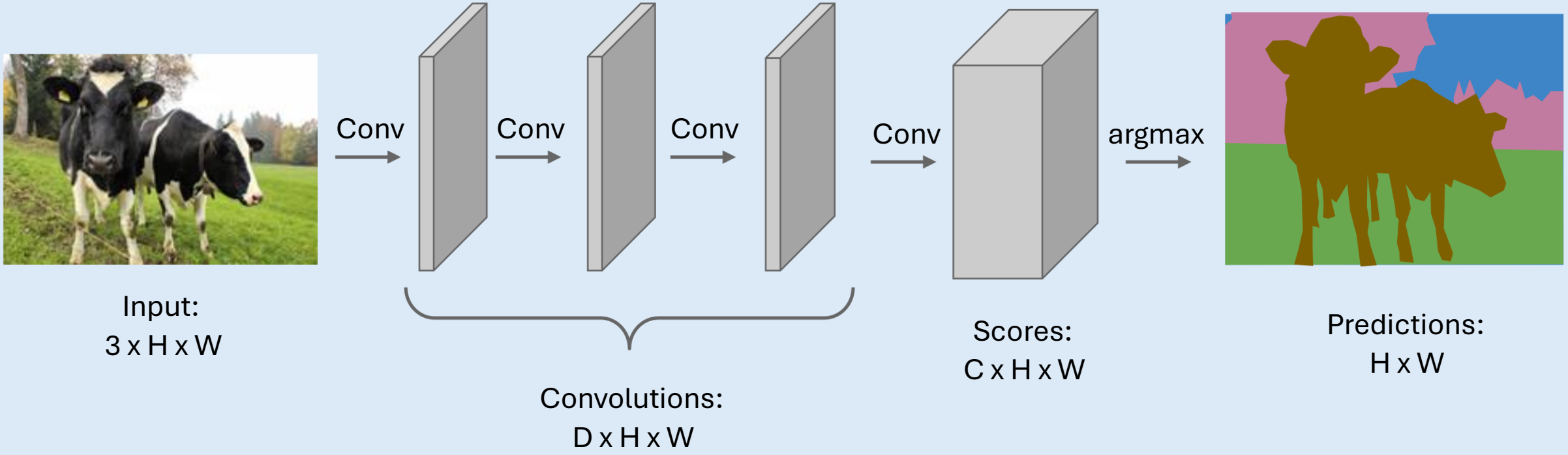


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

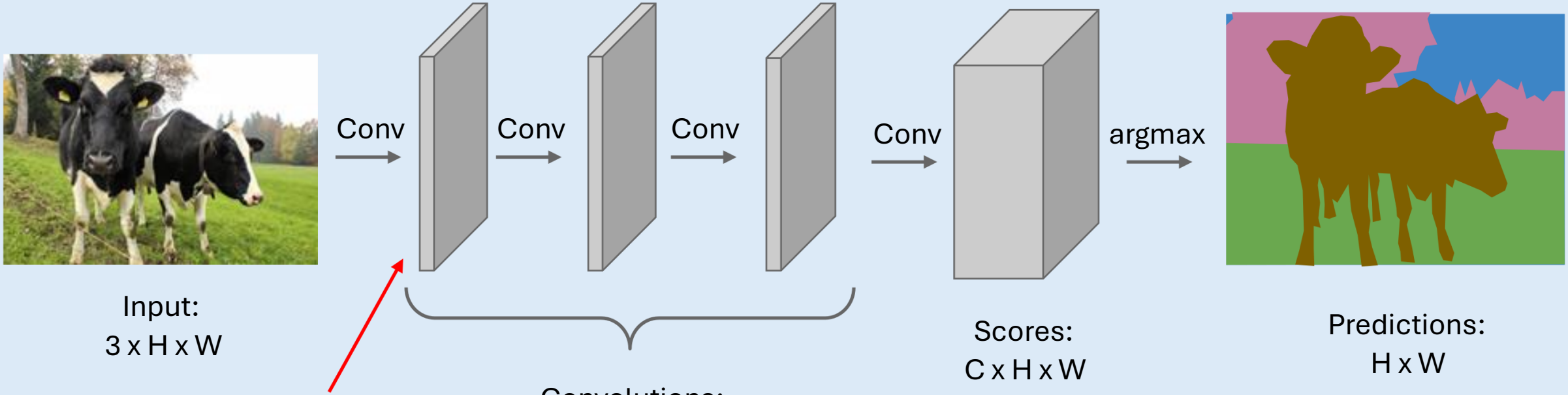
Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



Semantic Segmentation Idea: Fully Convolutional

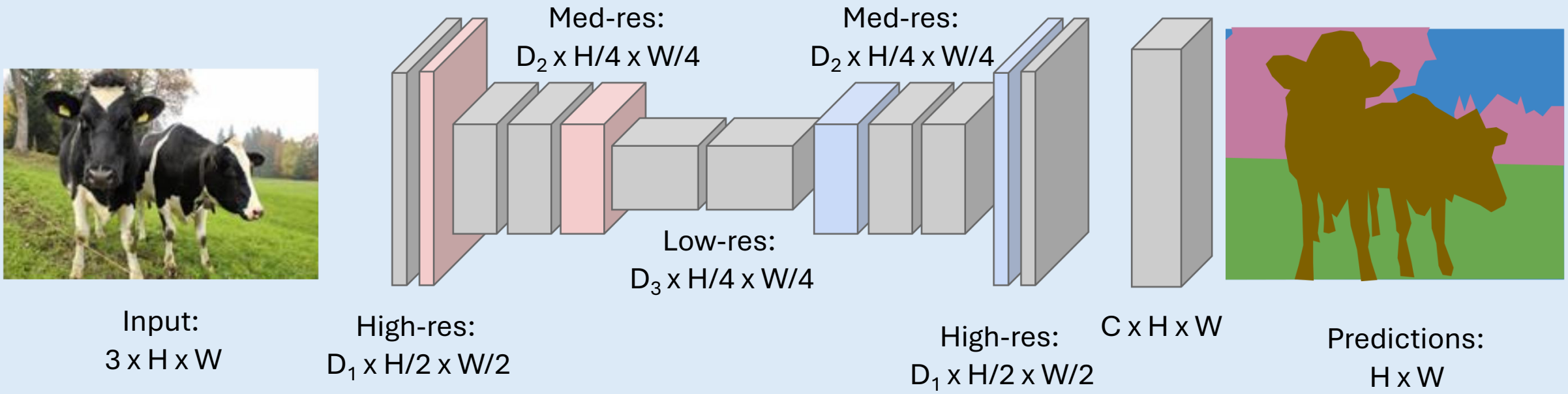
Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



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

Semantic Segmentation Idea: Fully Convolutional

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



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

Semantic Segmentation Idea: Fully Convolutional

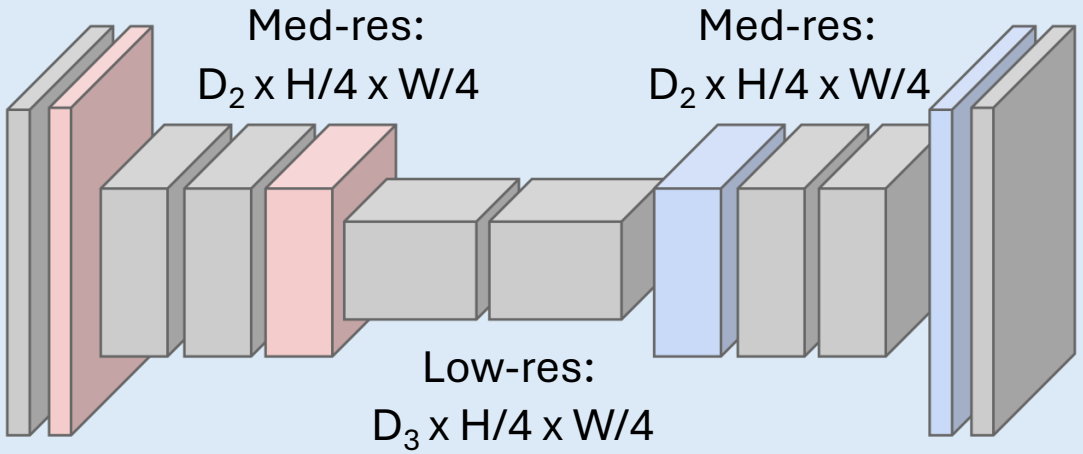
Downsampling:
Pooling, strided convolution

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

Upsampling:
???

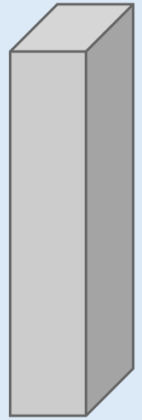


Input:
3 x H x W



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

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



C 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

In-Network upsampling: “Unpooling”

Nearest Neighbor

1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

“Bed of Nails”

1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

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

Input: 4 x 4

5	6
7	8

Output: 2 x 2

…

Rest of the network

Max Unpooling
Use positions from pooling layer

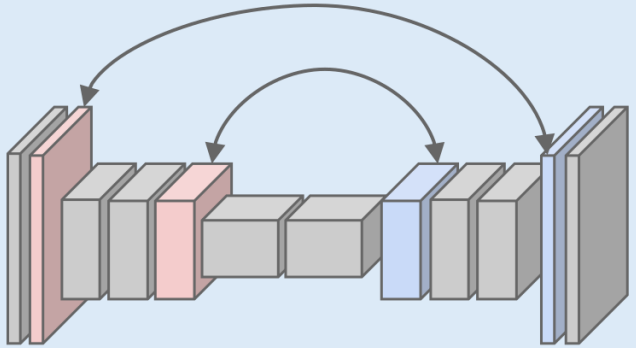
1	2
3	4

Input: 2 x 2

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

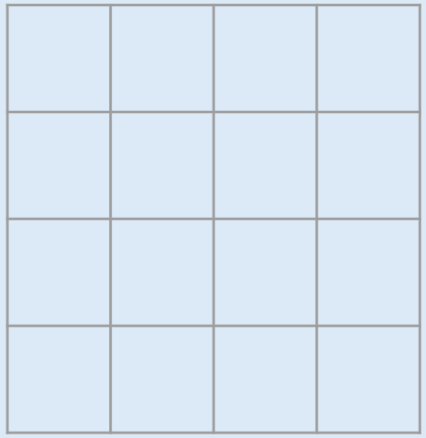
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

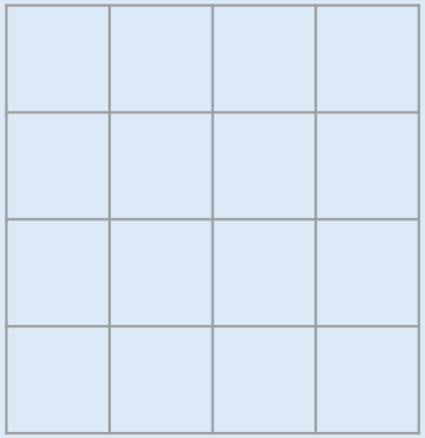


Learnable Upsampling

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



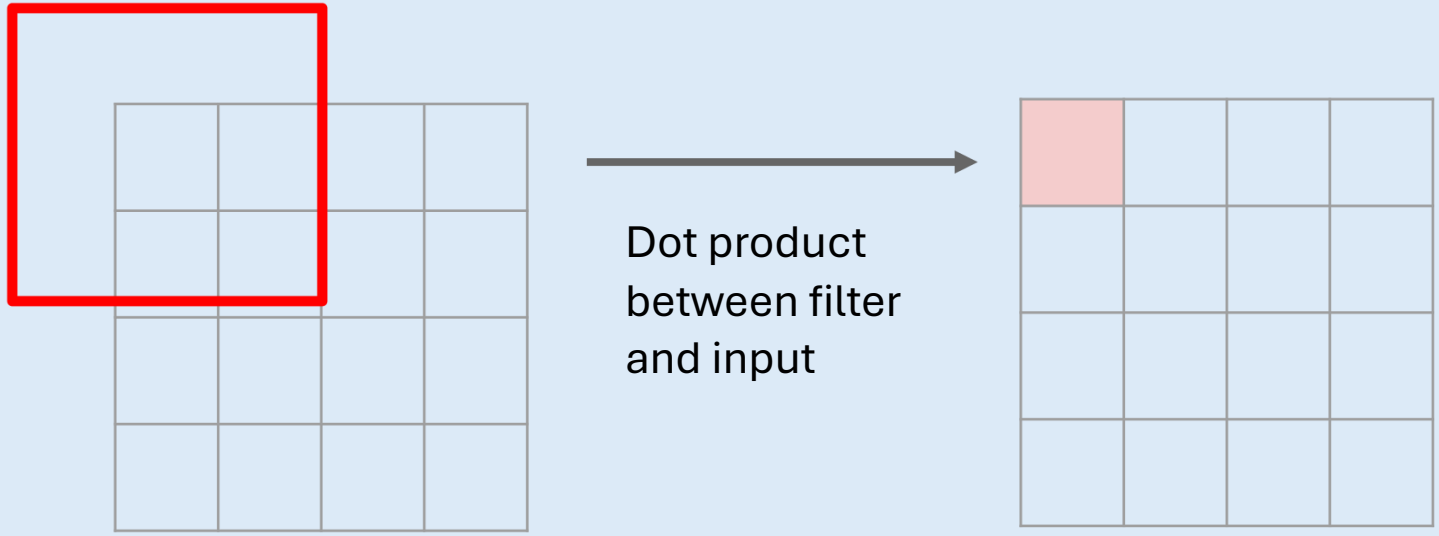
Input: 4 x 4



Output: 4 x 4

Learnable Upsampling

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

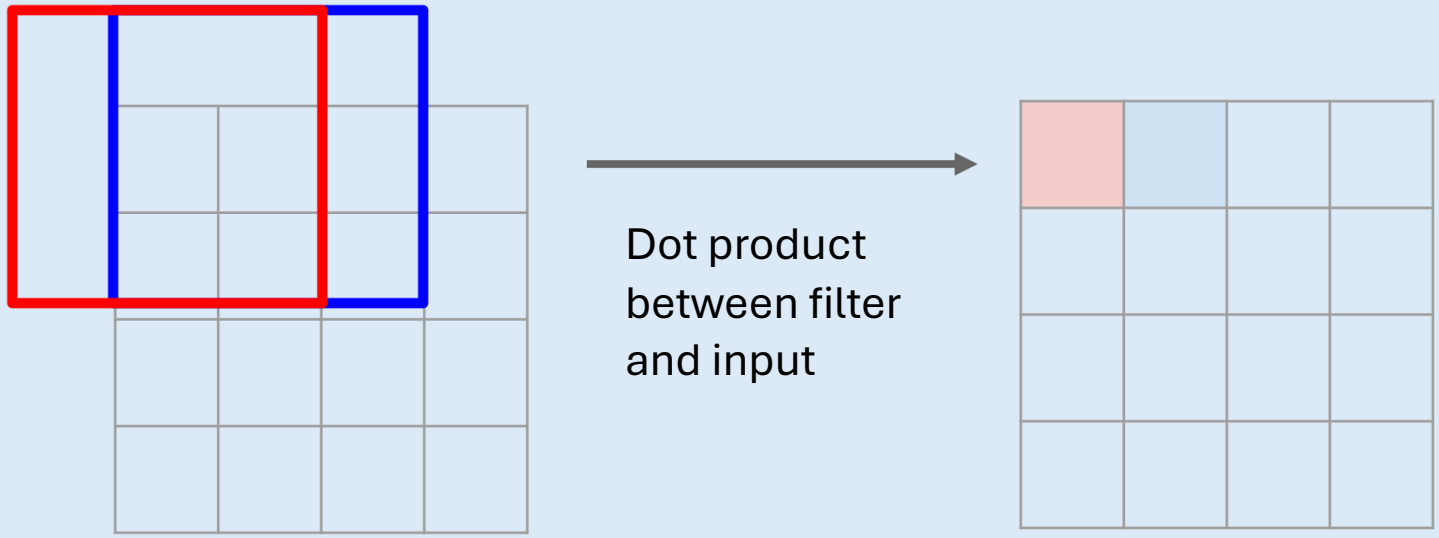


Input: 4 x 4

Output: 4 x 4

Learnable Upsampling

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

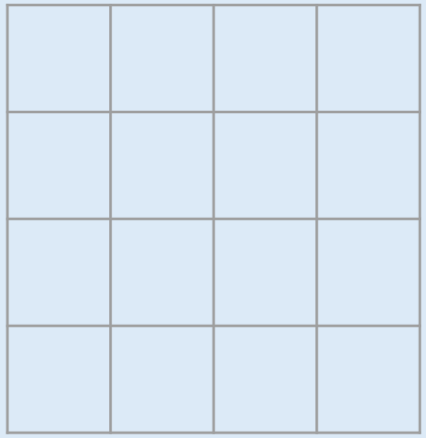


Input: 4 x 4

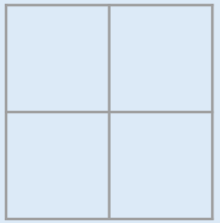
Output: 4 x 4

Learnable Upsampling

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



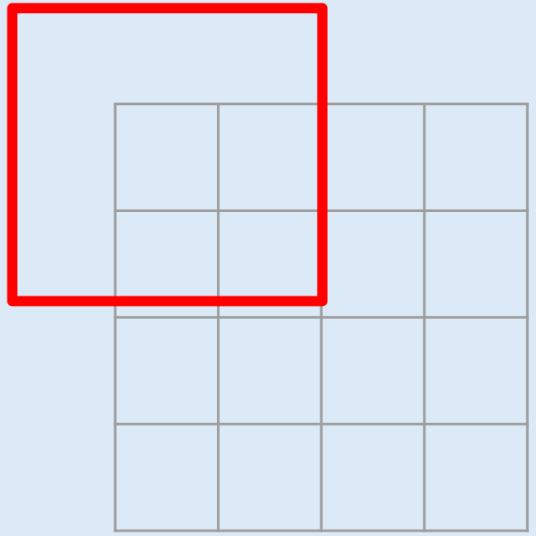
Input: 4 x 4



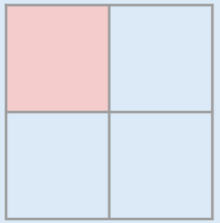
Output: 2 x 2

Learnable Upsampling

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



Dot product between filter and input

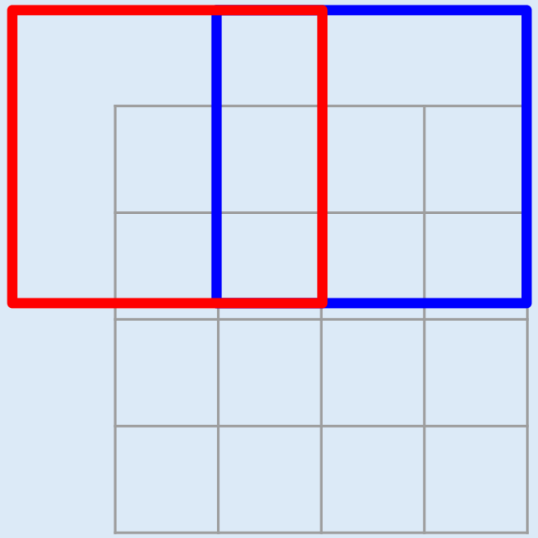


Input: 4 x 4

Output: 2 x 2

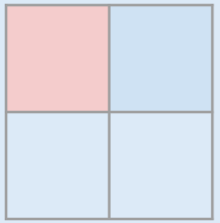
Learnable Upsampling

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



Input: 4 x 4

Dot product between filter and input



Output: 2 x 2

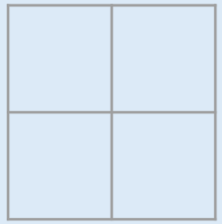
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

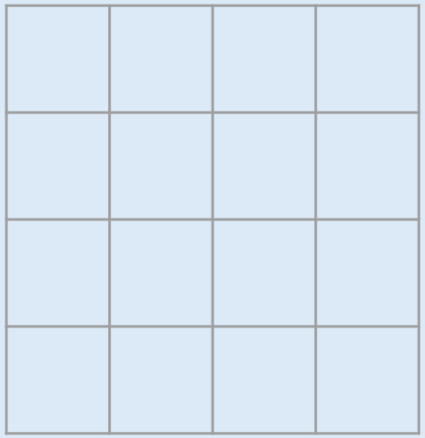
We can interpret strided convolution as “learnable downsampling”.

Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1



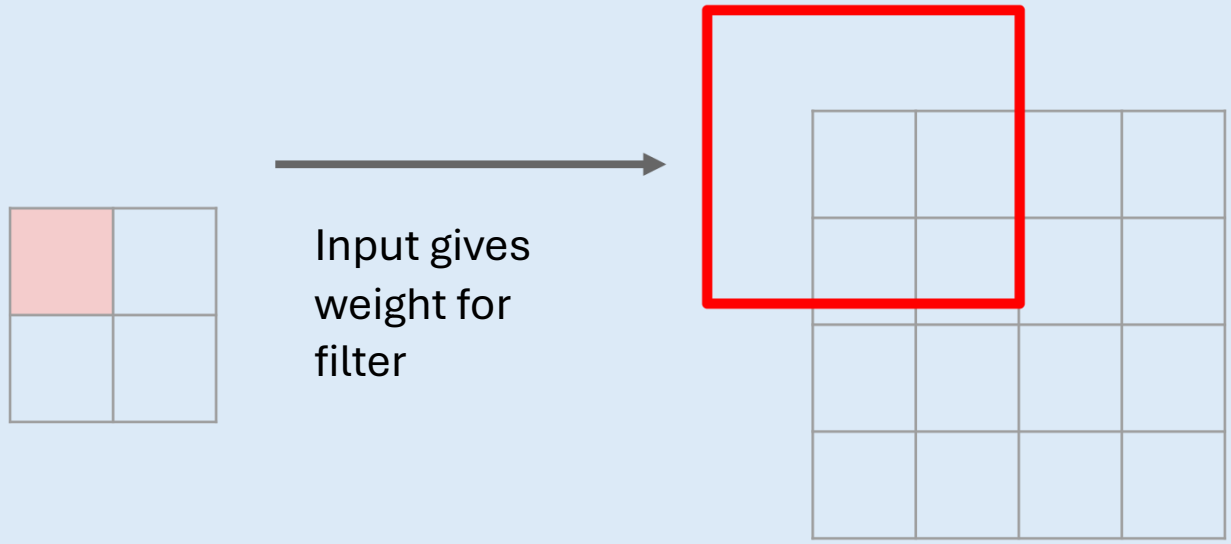
Input: 2 x 2



Output: 4 x 4

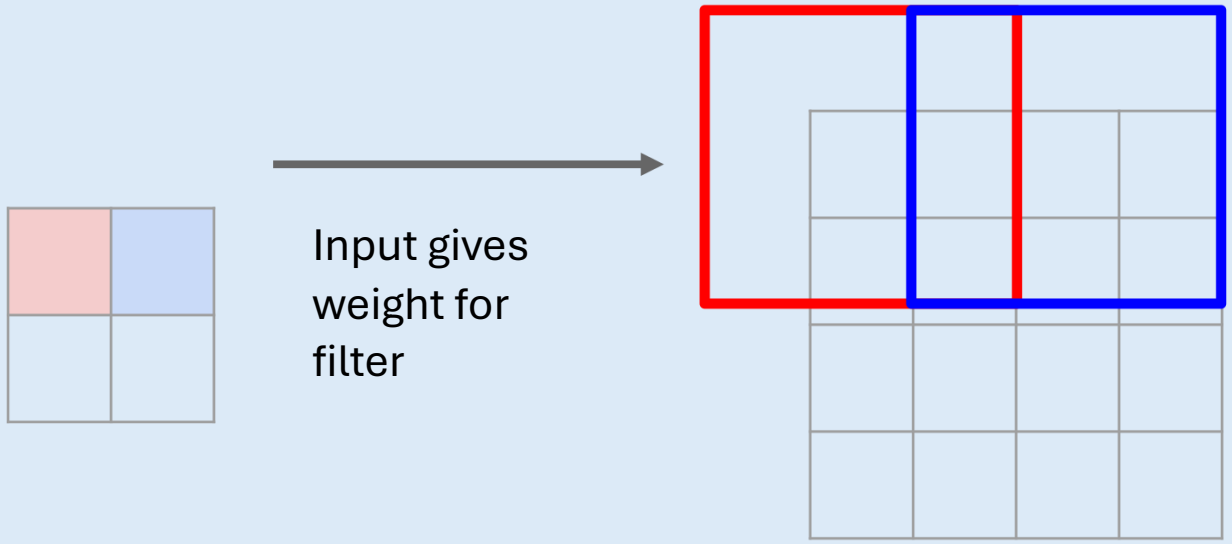
Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1



Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1

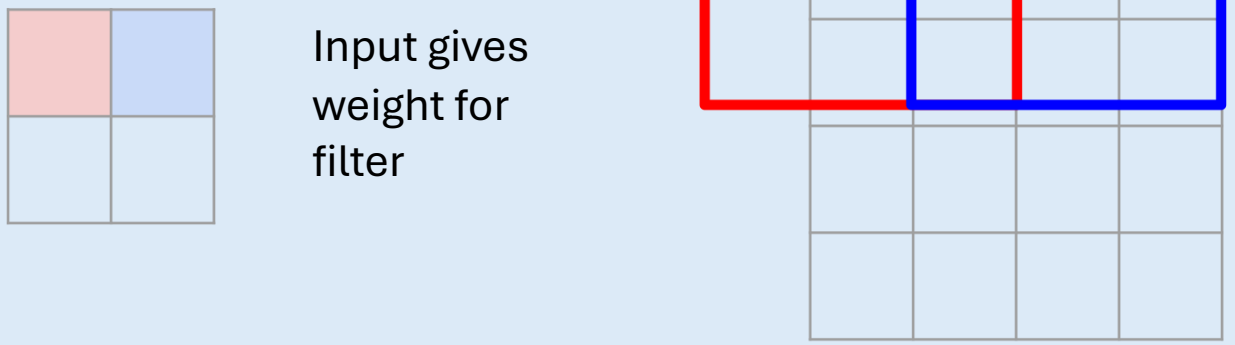


Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1



Input gives weight for filter

Sum where output overlaps

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

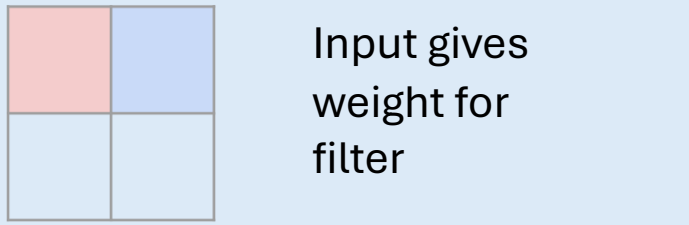
Stride gives ratio between movement in output and input

Input: 2 x 2

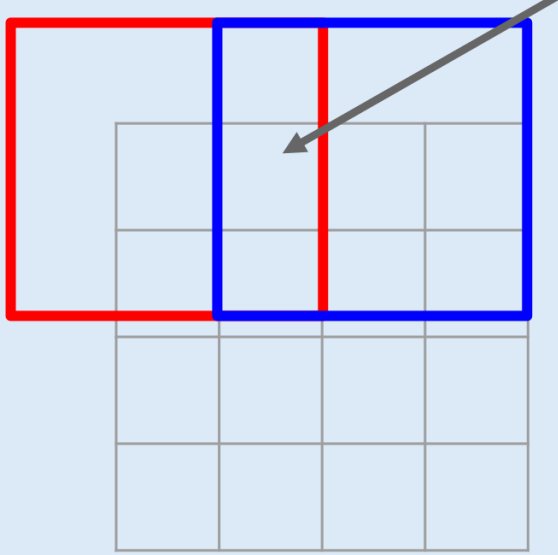
Output: 4 x 4

Learnable Upsampling: Transposed Convolution

3 x 3 transposed convolution, stride 2 pad 1



Input gives weight for filter



Sum where output overlaps

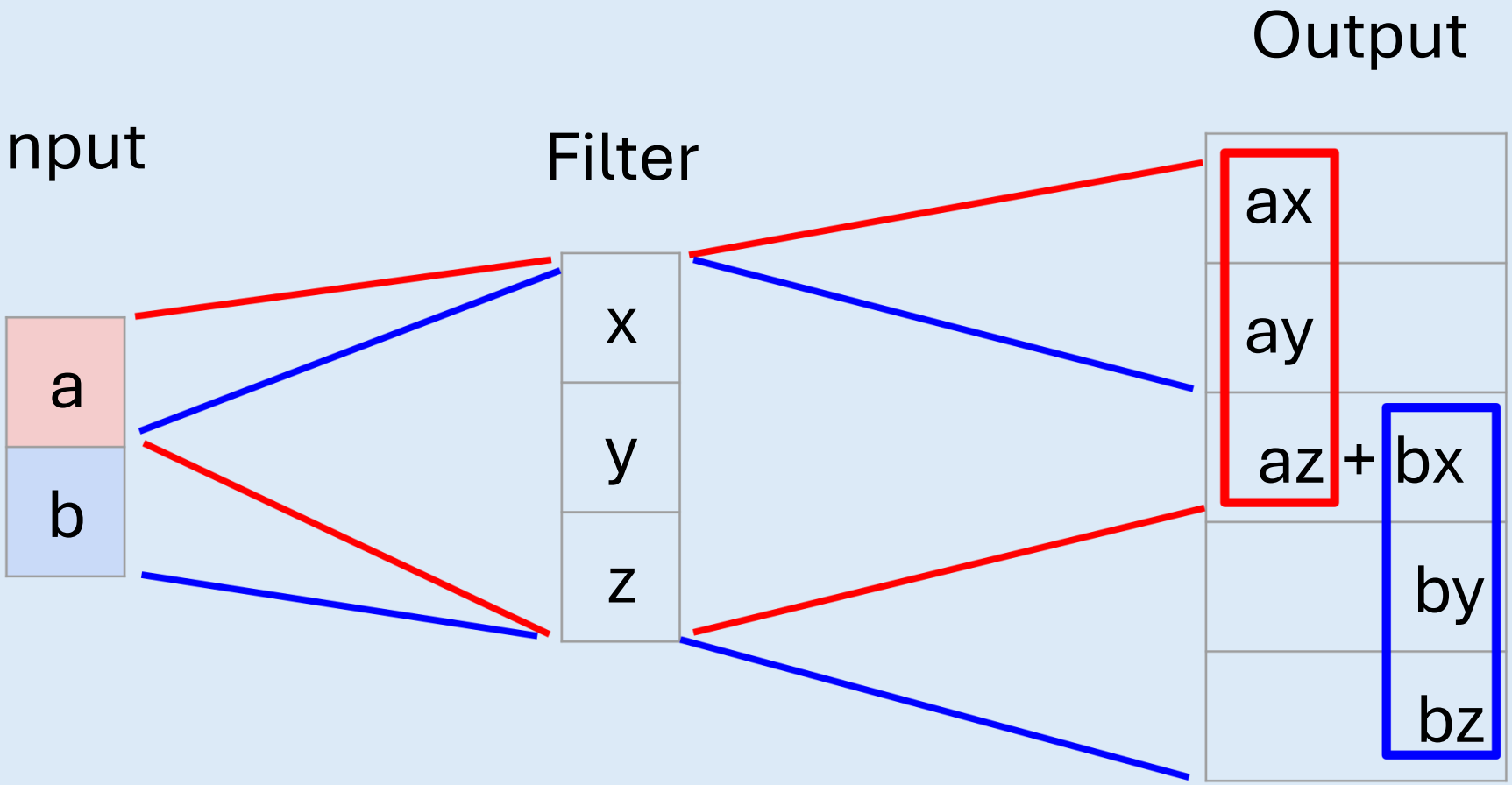
Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Input: 2 x 2

Output: 4 x 4

Learnable Upsampling: 1D Example



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

Learnable Upsampling: 2D Example

2 × 2 input, 3 × 3 kernel, stride 2, padding 0

Input X

1	2
3	4

Kernel K

1	2	1
2	4	2
1	2	1

First contribution: 1 × K

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

$$Y_{2i+u,2j+v} \mathrel{+}= X_{i,j} K_{u,v}$$

Each input value scales the same kernel.

Place its top-left corner at (2i, 2j), then add to the output.

One channel • no bias • example kernel values

2D Example: Place Weighted Kernels

Move each 3 × 3 patch by 2 output cells per input step.

1 × K at (0, 0)

1	2	1		
2	4	2		
1	2	1		

2 × K at (0, 2)

		2	4	2
		4	8	4
		2	4	2

3 × K at (2, 0)

3	6	3		
6	12	6		
3	6	3		

4 × K at (2, 2)

		4	8	4
		8	16	8
		4	8	4

All four grids share the same output coordinates.

Add them element by element; blank cells contribute 0.

2D Example: Sum Overlapping Values

2 × 2 input becomes a 5 × 5 output.

Output Y

1	2	3	4	2
2	4	6	8	4
4	8	10	12	6
6	12	14	16	8
3	6	7	8	4

Center cell (zero-based indices)

Four kernel patches overlap here.

$$Y_{2,2} = X_{0,0}K_{2,2} + X_{0,1}K_{2,0} + X_{1,0}K_{0,2} + X_{1,1}K_{0,0}$$

$$= 1\cdot1 + 2\cdot1 + 3\cdot1 + 4\cdot1 = 10$$

Overlapping contributions are summed.

The same rule applies to every output cell.

No averaging or interpolation step is needed.

2D Example: Padding and Output Size

For dilation 1, in each spatial dimension:

$$H_{out} = (H_{in} - 1)s - 2p + k + o$$

p = 0, o = 0

1	2	3	4	2
2	4	6	8	4
4	8	10	12	6
6	12	14	16	8
3	6	7	8	4

5 × 5

p = 1, o = 0

4	6	8
8	10	12
12	14	16

3 × 3

p = 1, o = 1

4	6	8	4
8	10	12	6
12	14	16	8
6	7	8	4

4 × 4

p: padding o: output_padding (selects the output size)

Semantic Segmentation Idea: Fully Convolutional

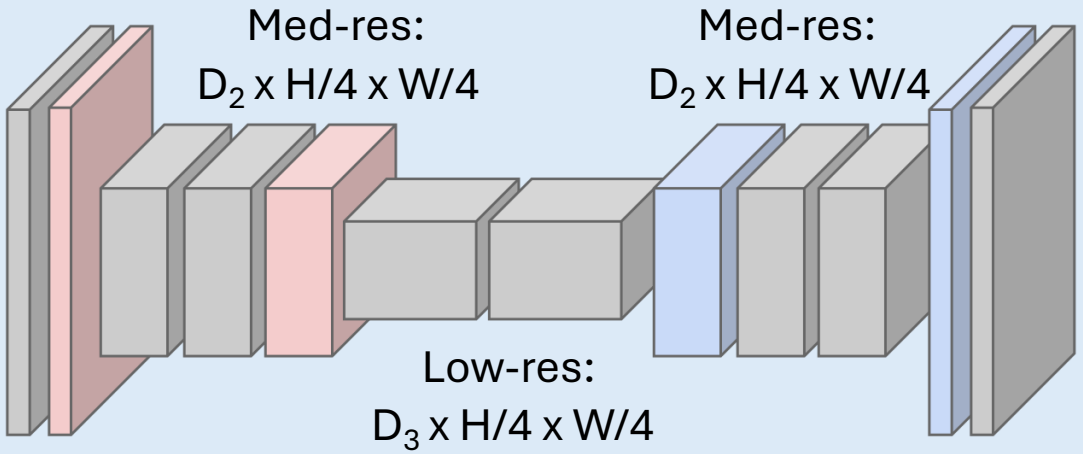
Downsampling:
Pooling, strided convolution

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

Upsampling:
Unpooling or strided transposed convolution



Input:
3 x H x W



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

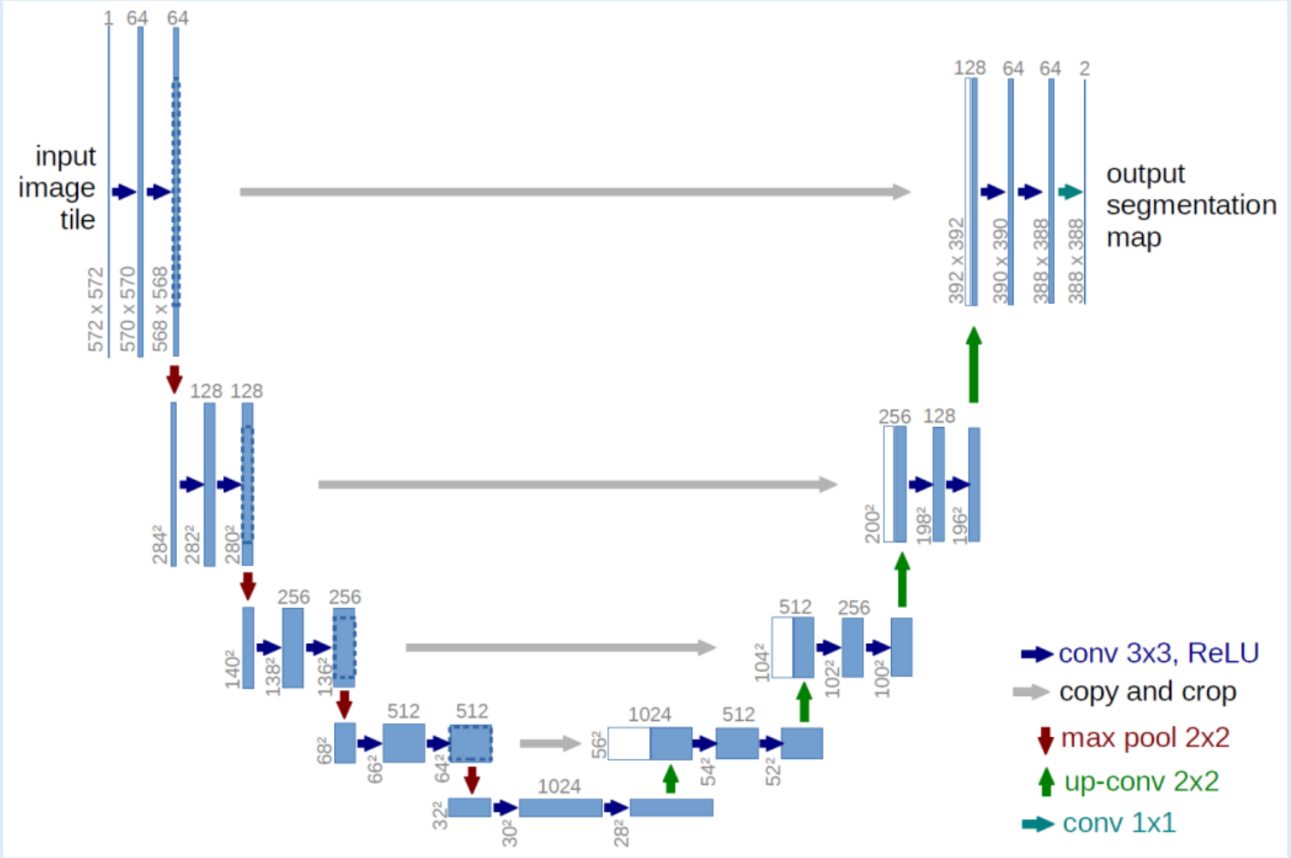
High-res:
D_1 x H/2 x W/2



Predictions:
H x W

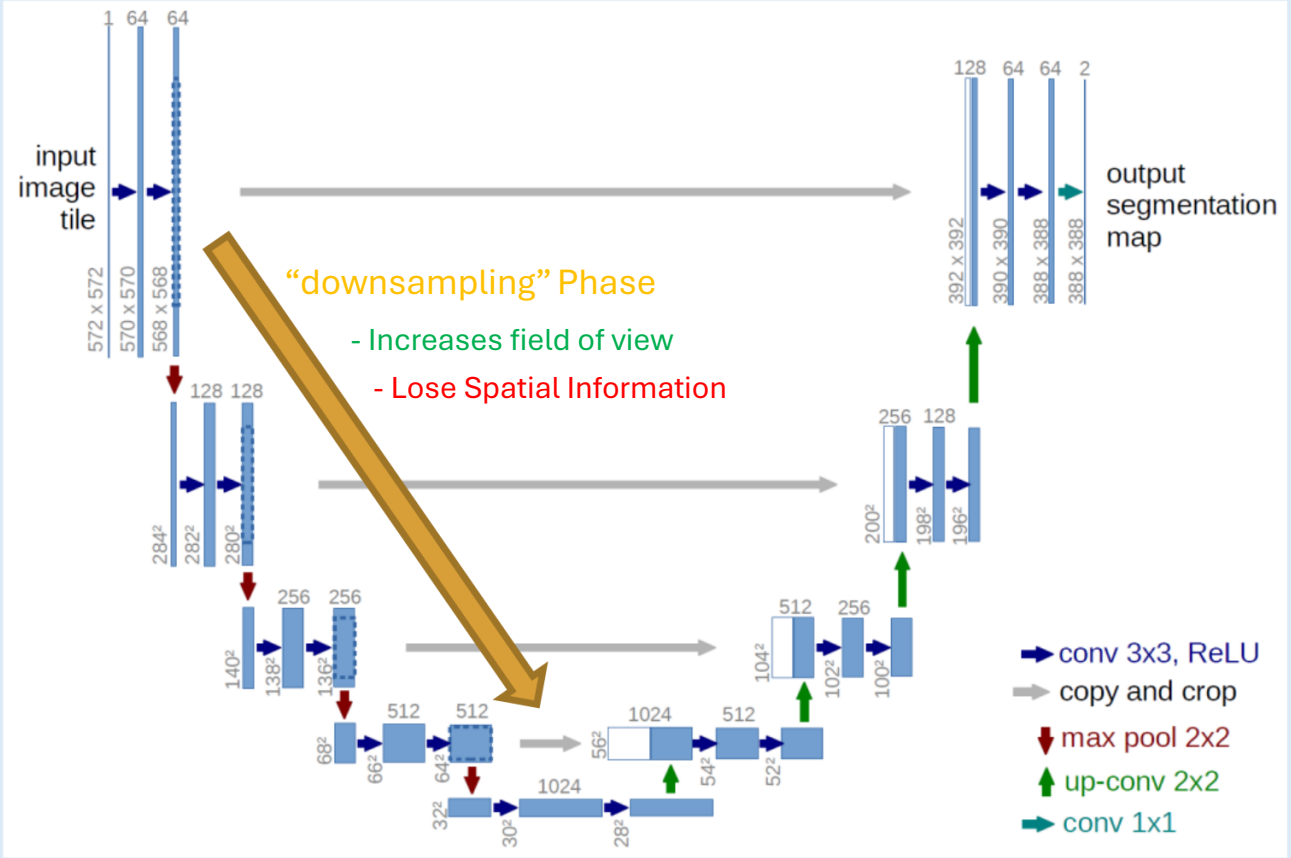
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

U-Net



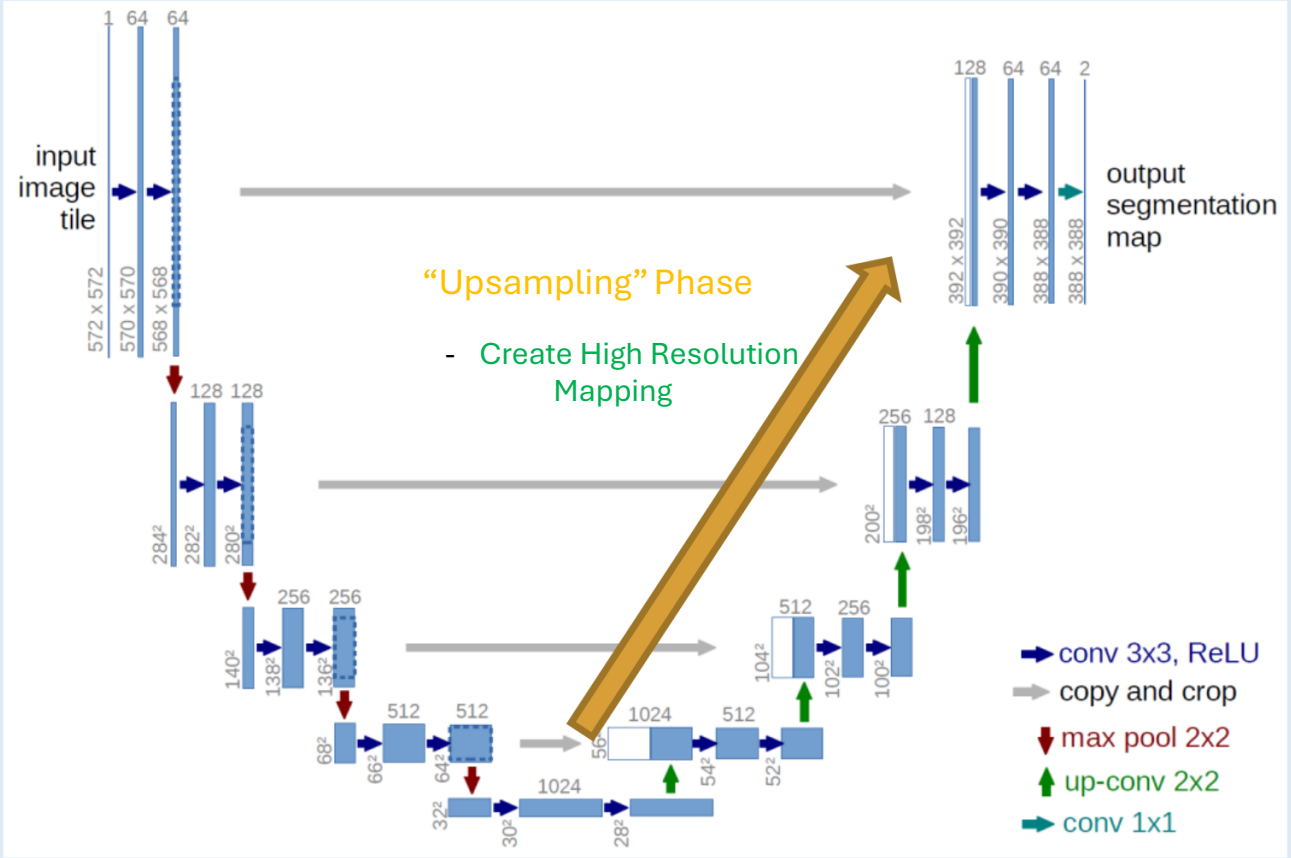
Ronneberger et al. (2015) U-net Architecture

U-Net



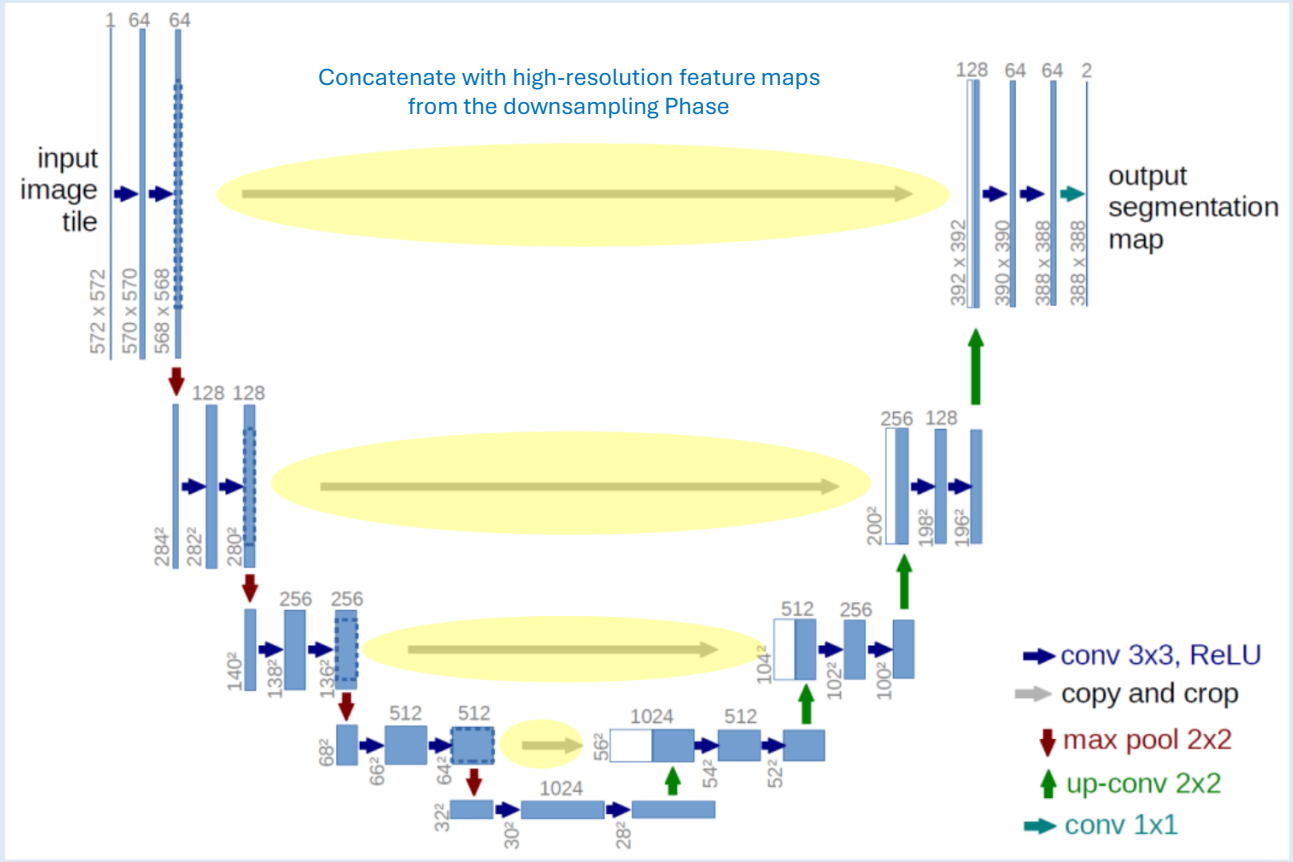
Ronneberger et al. (2015) U-net Architecture

U-Net



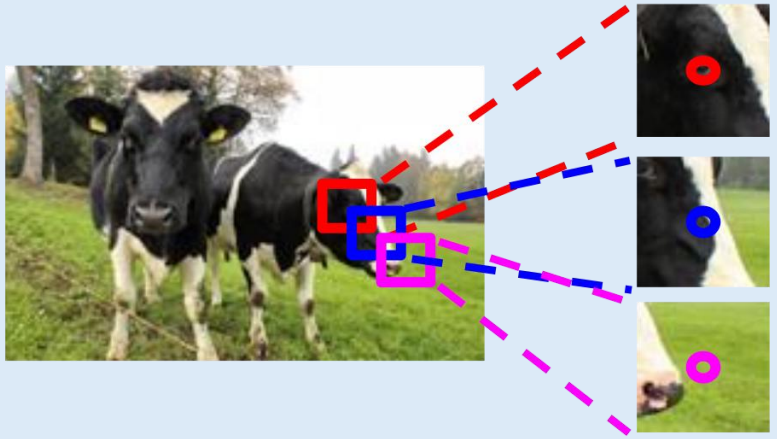
Ronneberger et al. (2015) U-net Architecture

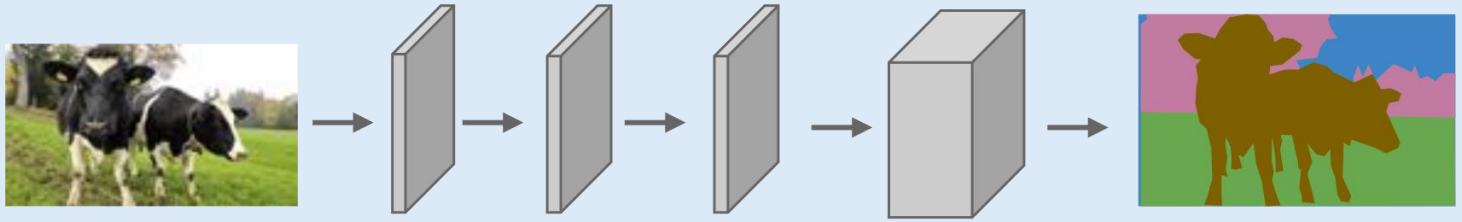
U-Net



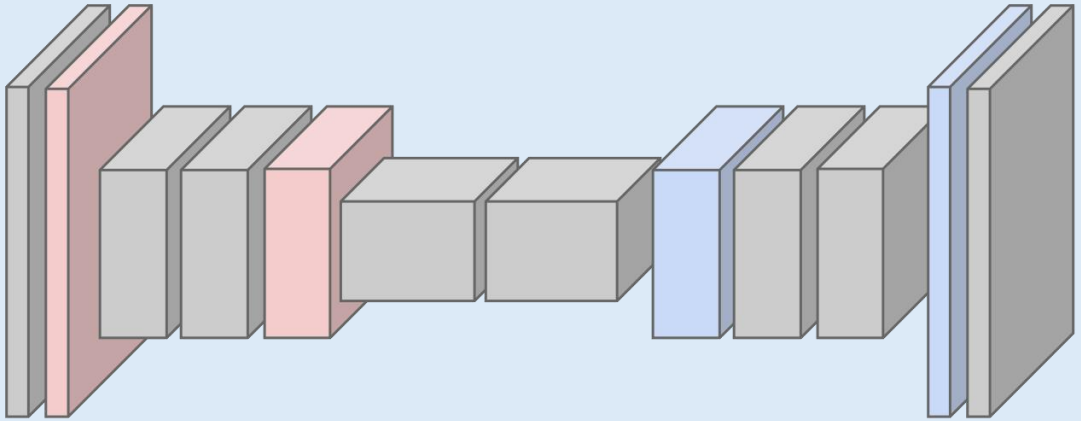
Ronneberger et al. (2015) U-net Architecture

Semantic Segmentation: Summary











Semantic Segmentation

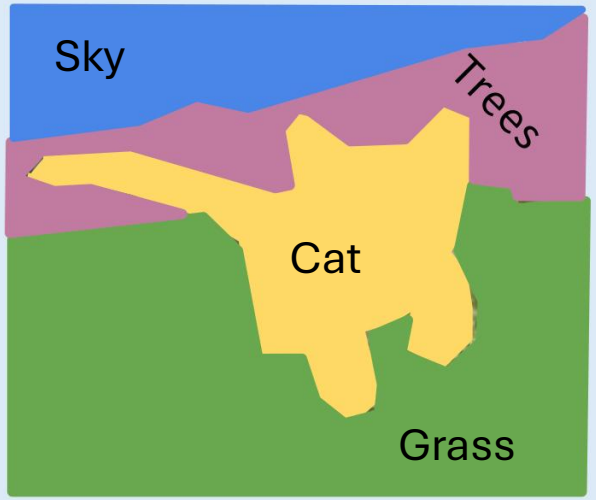
Label each pixel in the image with a category label

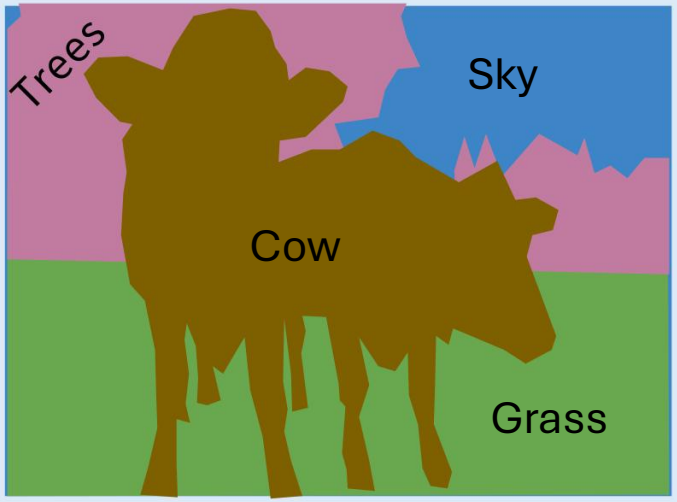
Don’t differentiate instances, only care about pixels



This image is CC0 public domain



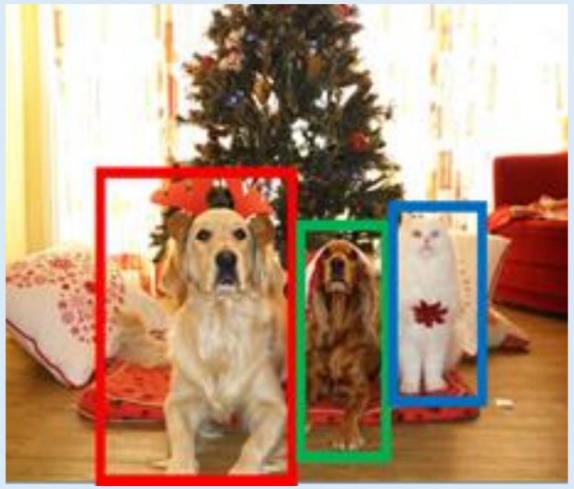




Object Detection

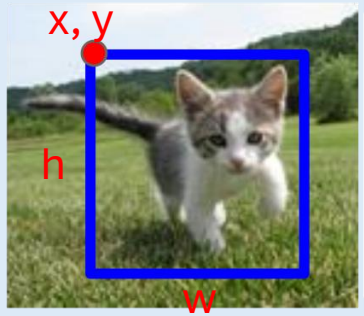
Classification	Semantic Segmentation	Object Detection	Instance Segmentation
			
CAT	GRASS, CAT, TREE, SKY	DOG, DOG, CAT	DOG, DOG, CAT
No spatial extent	No objects, just pixels	Multiple Object	

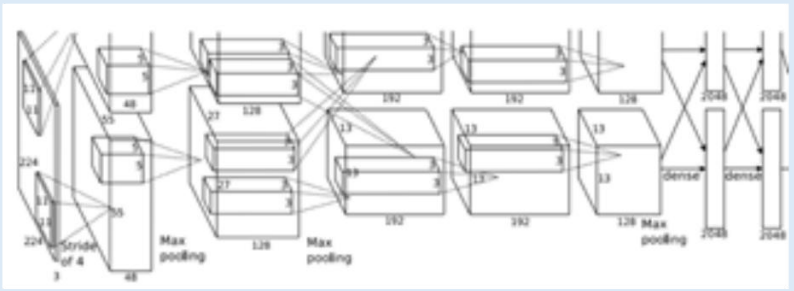
Object Detection

Classification	Semantic Segmentation	Object Detection	Instance Segmentation
			
CAT	GRASS, CAT, TREE, SKY	DOG, DOG, CAT	DOG, DOG, CAT
No spatial extent	No objects, just pixels	Multiple Object	

Object Detection: Single Object

(Classification + Localization)





This image is CC0 public domain

Fully Connected: 4096 to 1000

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

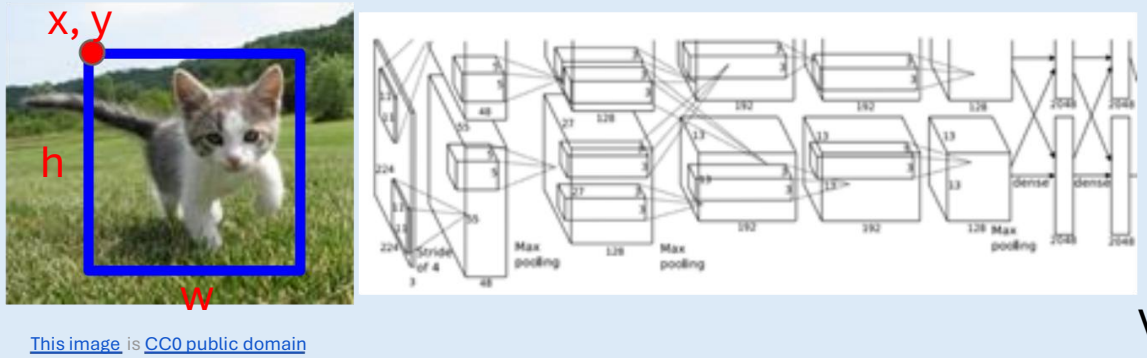
Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

Object Detection: Single Object

(Classification + Localization)



This image is CC0 public domain

Correct label:
Cat

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

Softmax Loss

Fully Connected: 4096 to 1000

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

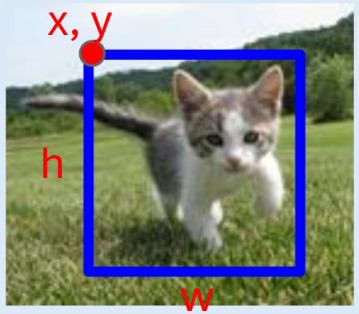
L2 Loss

Correct box:
(x', y', w', h')

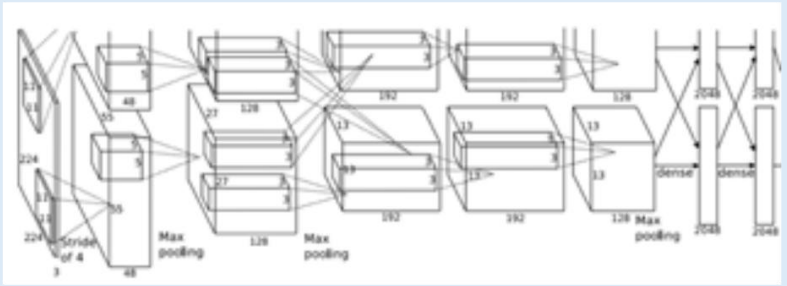
Treat localization as a regression problem!

Object Detection: Single Object

(Classification + Localization)



This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

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

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x', y', w', h')

Multitask Loss

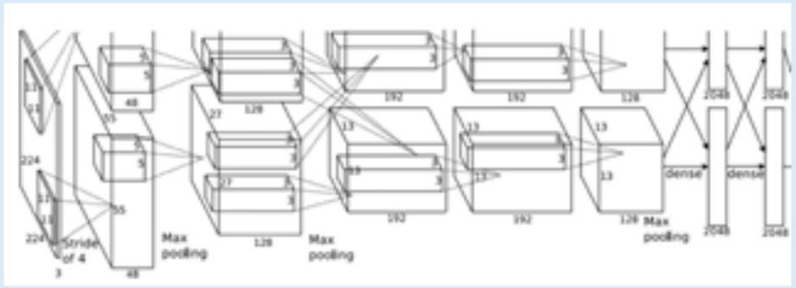
+

Loss

Treat localization as a regression problem!

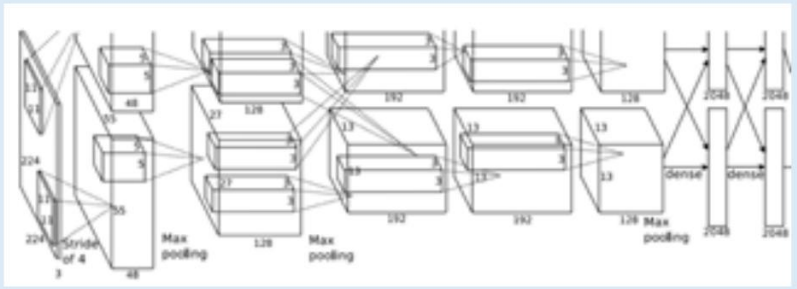
Object Detection: Multiple Objects





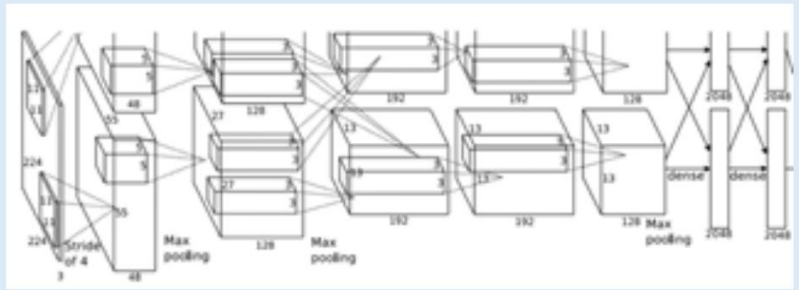
CAT: (x, y, w, h)





DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



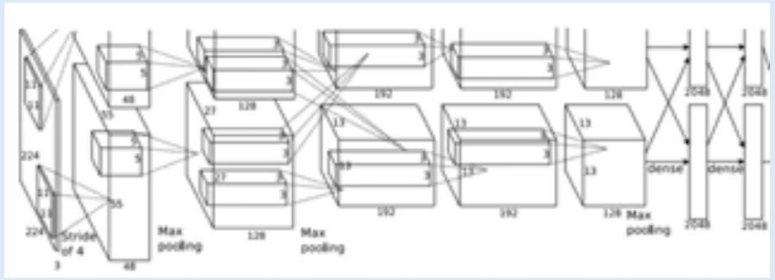


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Object Detection: Multiple Objects

Each image needs a different number of outputs!

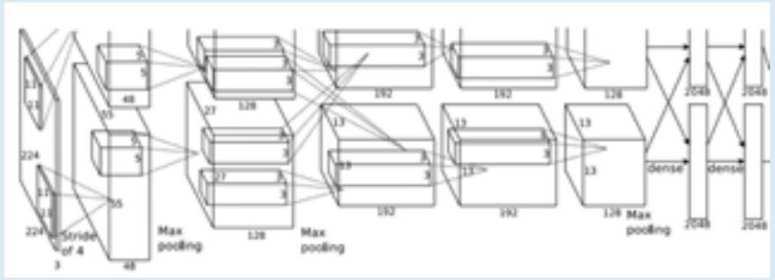




CAT: (x, y, w, h)

4 numbers

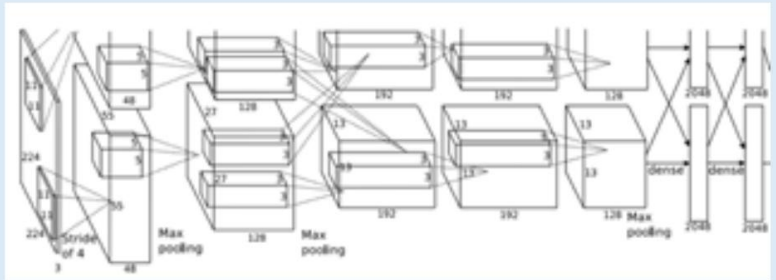




DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

12 numbers





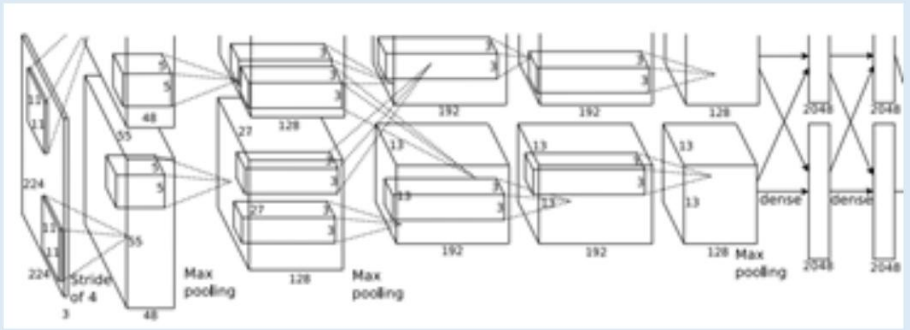
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many numbers!

Object Detection: Multiple Objects

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



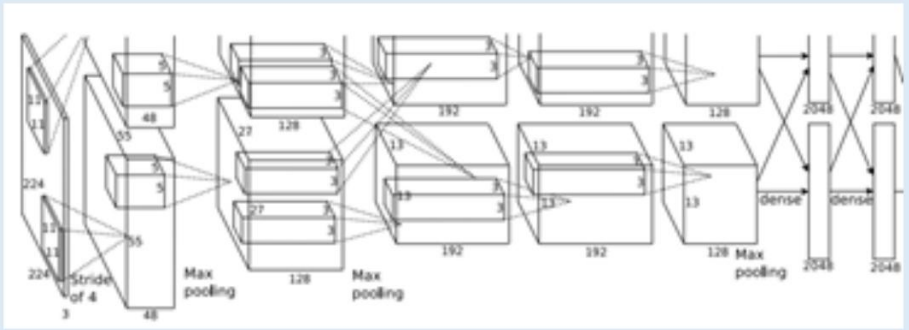


Dog? NO
Cat? NO
Background? YES

Object Detection: Multiple Objects

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



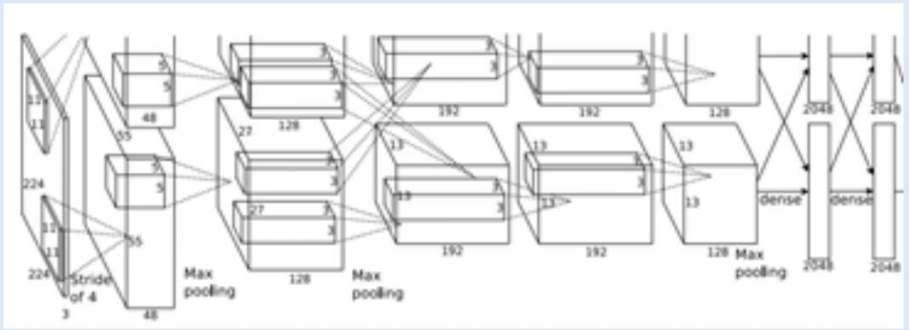


Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

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



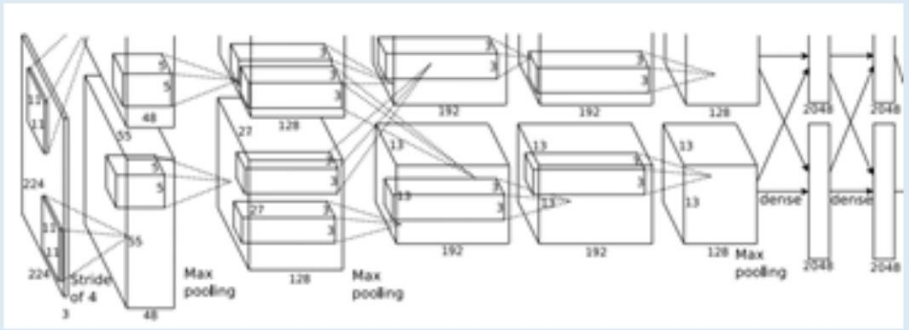


Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

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



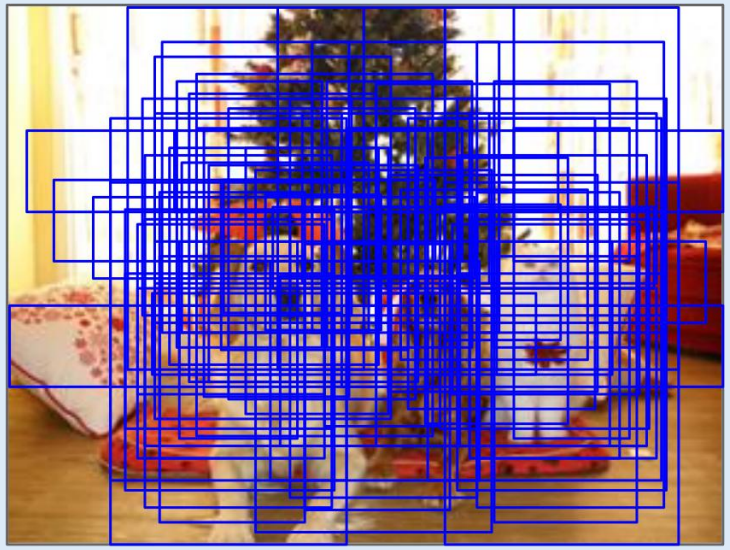


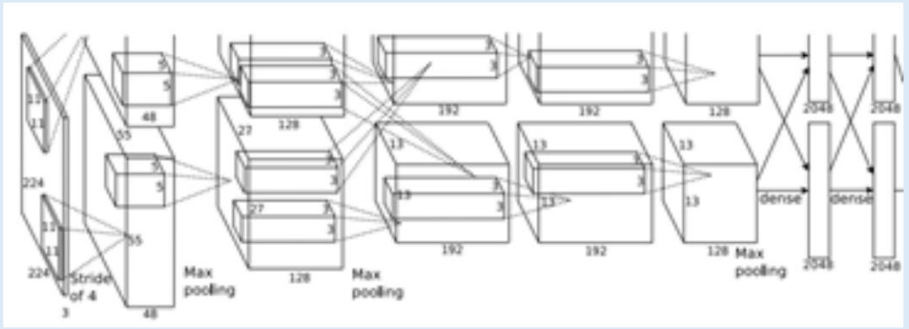
Dog? NO
Cat? YES
Background? NO

Q: What's the problem with this approach?

Object Detection: Multiple Objects

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



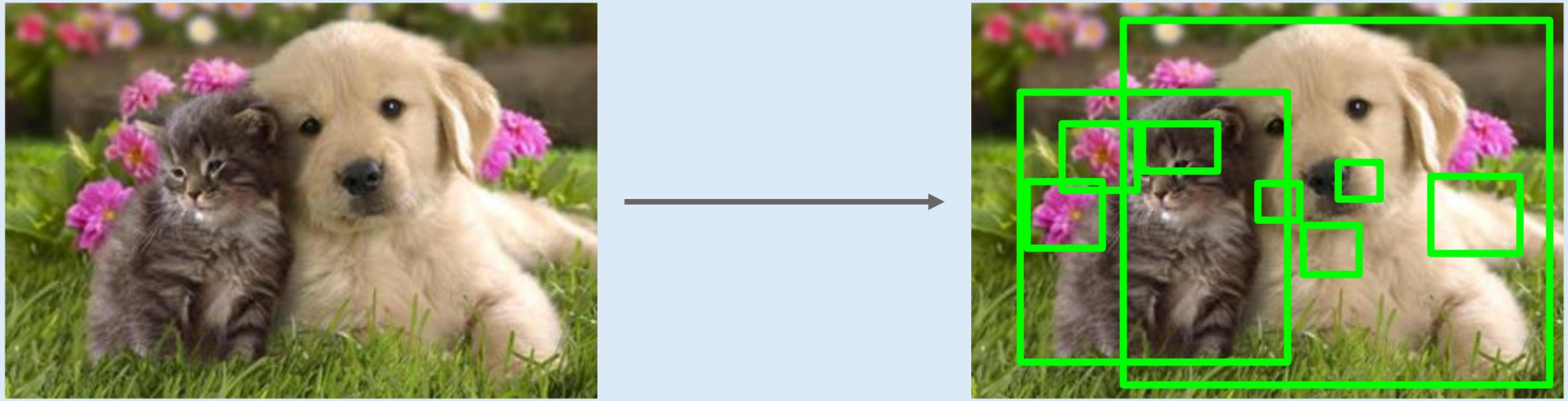


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Region Proposals: Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on 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



Input image

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015source. Reproduced with permission.

R-CNN

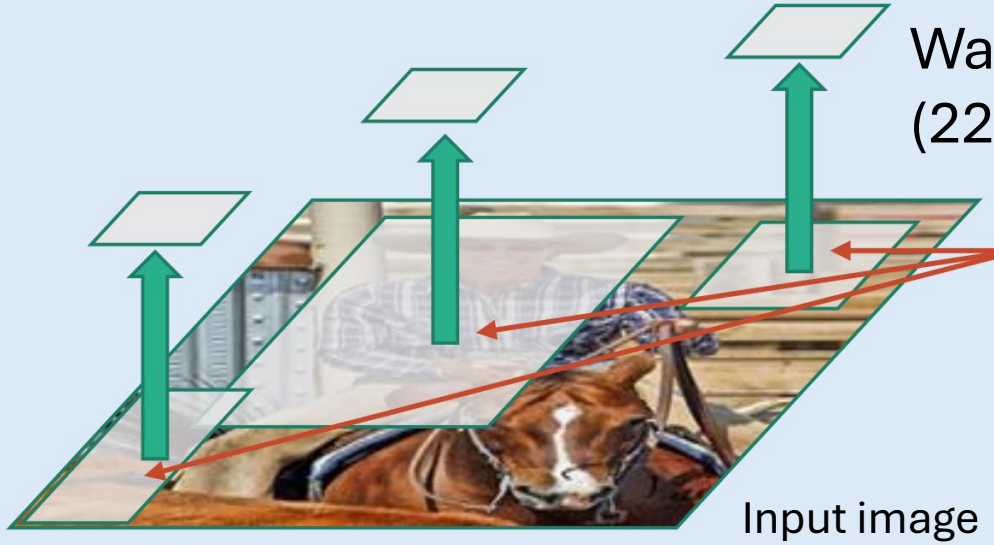


Input image

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

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015source. Reproduced with permission.

R-CNN



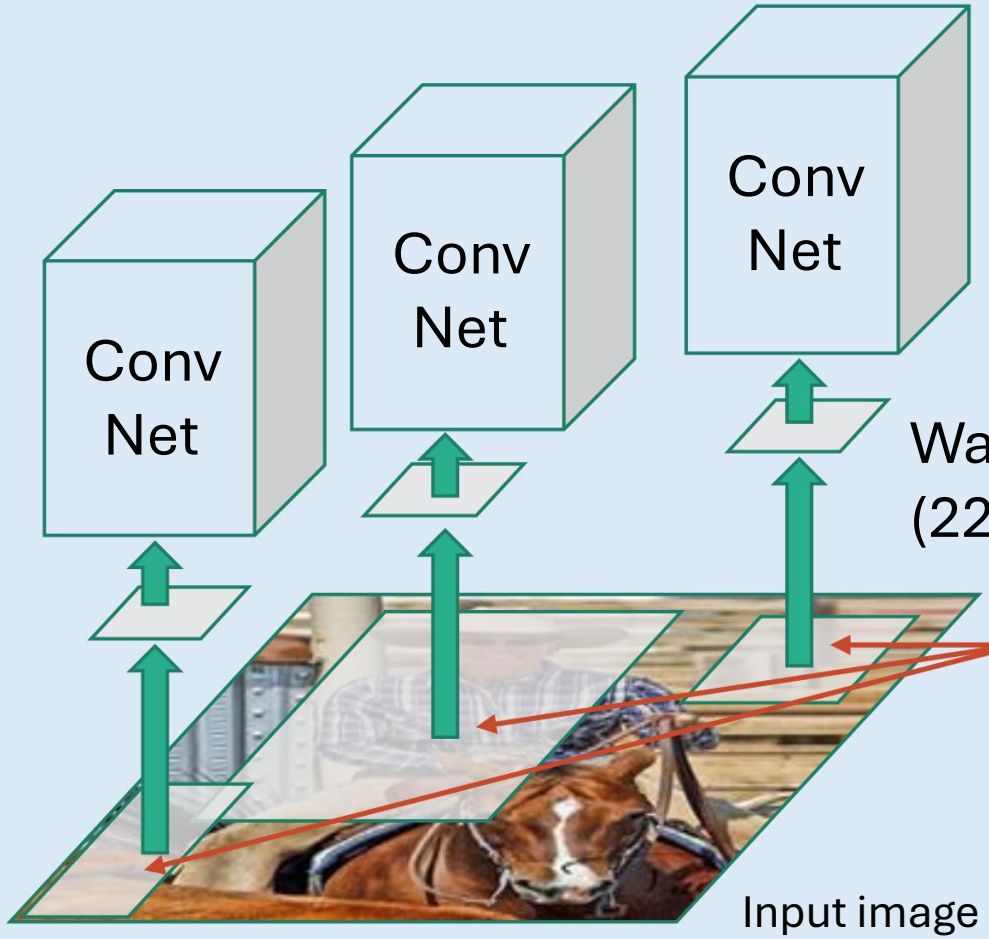
Warped image regions (224x224 pixels)

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

Input image

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015source. Reproduced with permission.

R-CNN



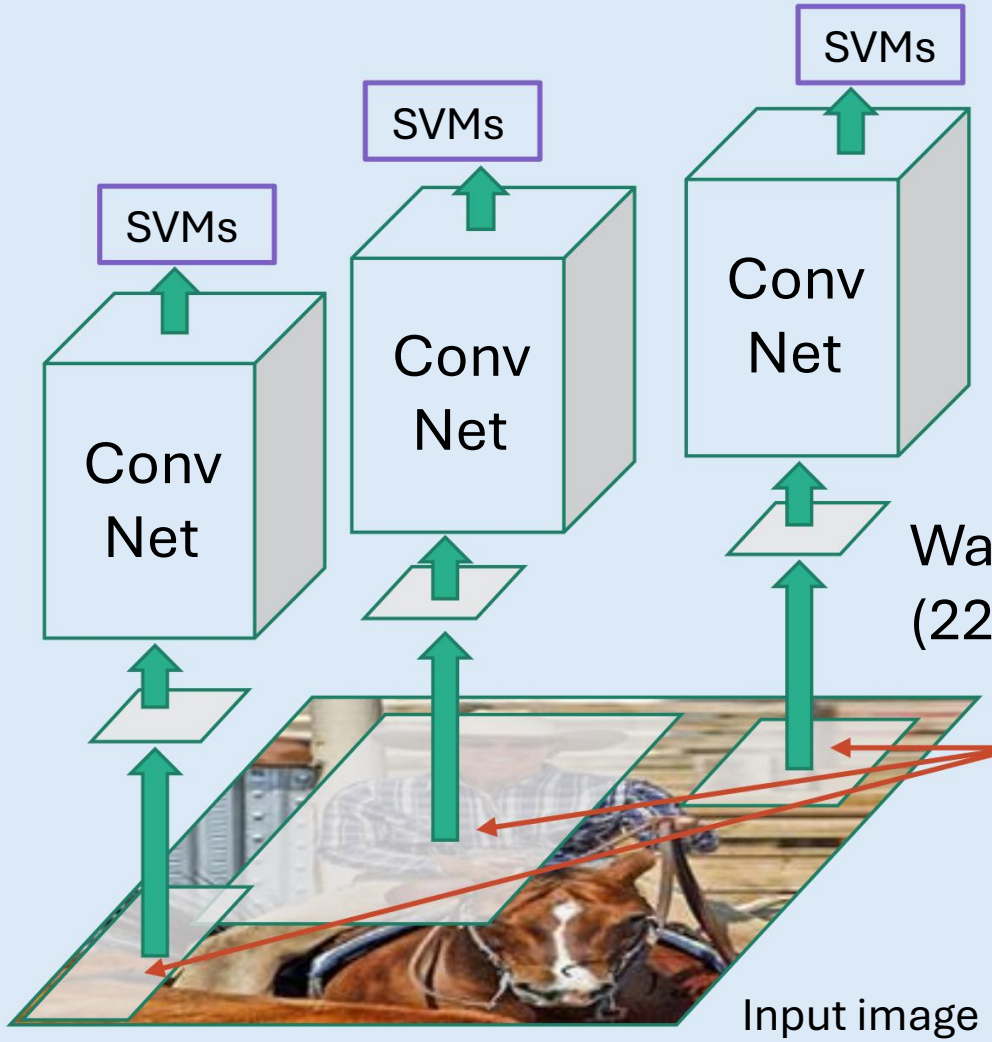
Forward each region through ConvNet (ImageNet-pretranied)

Warped image regions (224x224 pixels)

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

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015source. Reproduced with permission.

R-CNN



Classify regions with SVMs

Forward each region through ConvNet (ImageNet-pretranied)

Warped image regions (224x224 pixels)

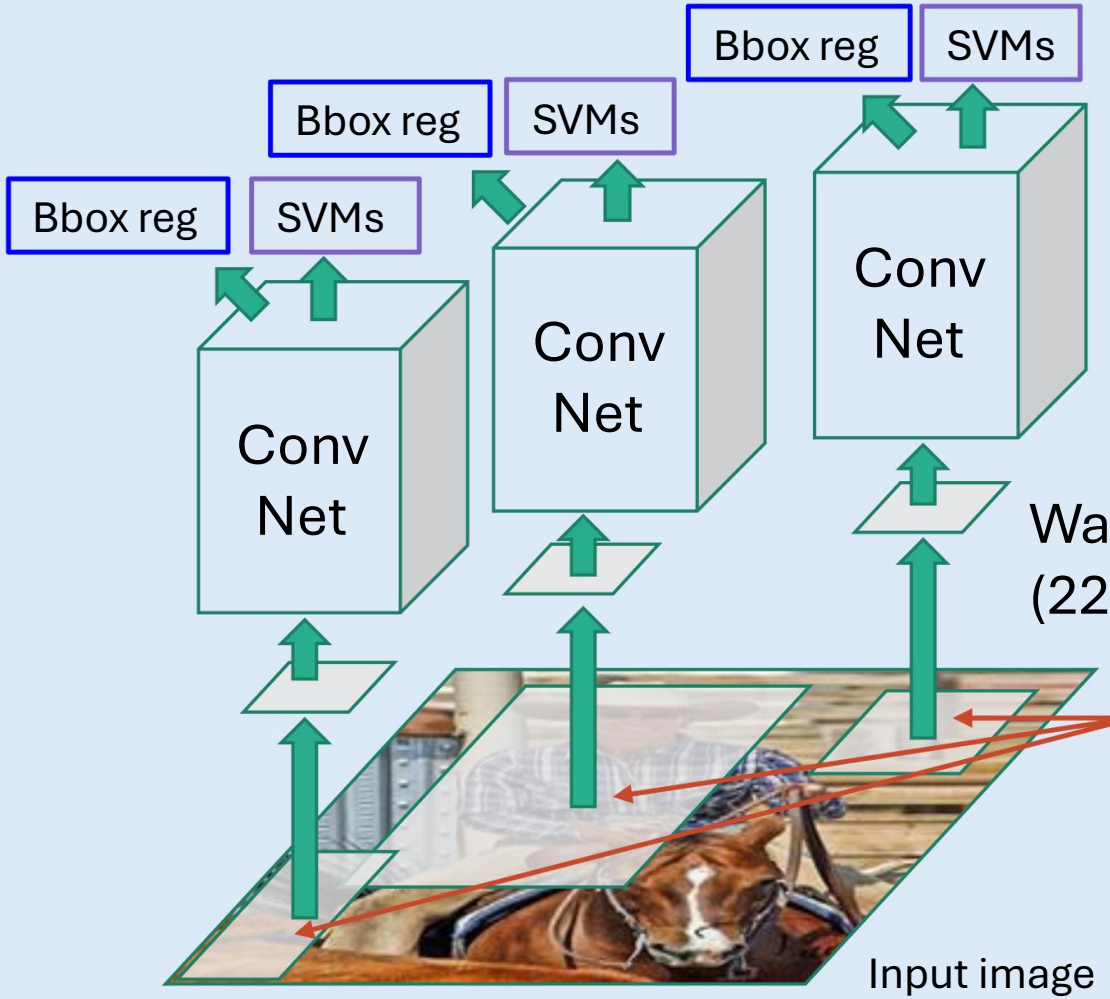
Regions of Interest (RoI) from a proposal method (~2k)

Input image

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015source. Reproduced with permission.

R-CNN

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



Classify regions with SVMs

Forward each region through ConvNet (ImageNet-pretranied)

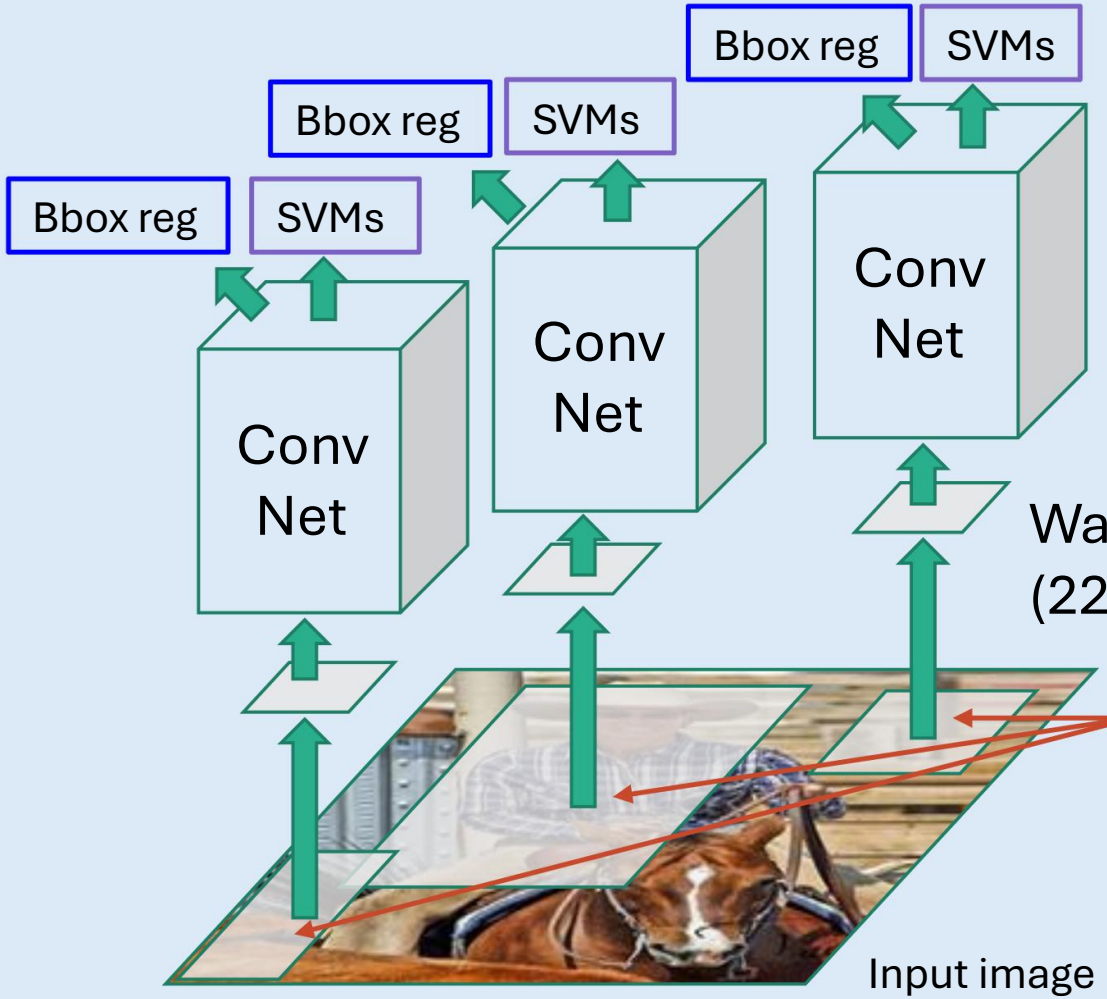
Warped image regions (224x224 pixels)

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

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015source. Reproduced with permission.

R-CNN

Predict "corrections" to the RoI: 4 numbers: (dx, dy, dw, dh)



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

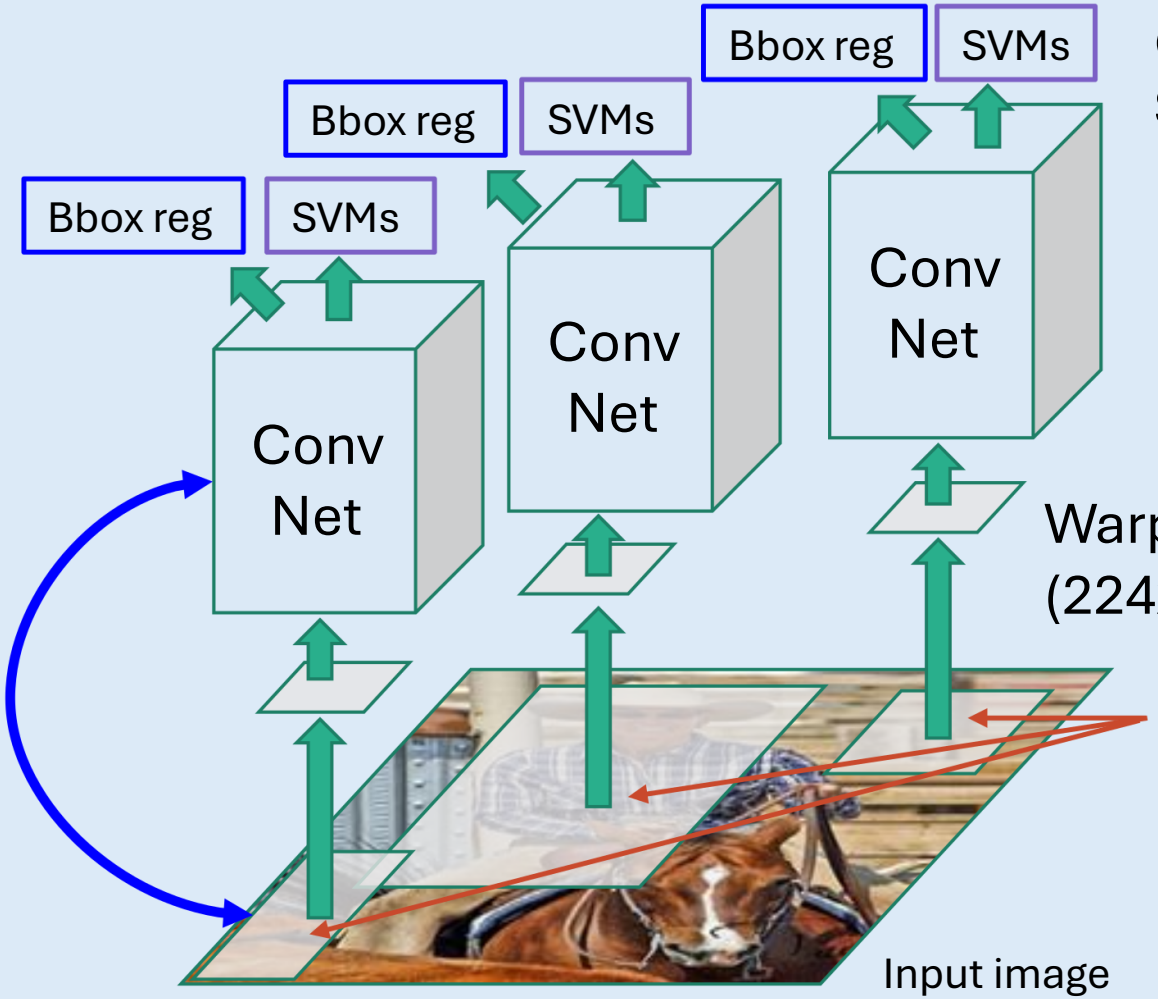
Regions of Interest (RoI) from a proposal method (~2k)

Problem: Very slow! Need to do ~2k independent forward passes for each image!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015source. Reproduced with permission.

“Slow” R-CNN

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



Classify regions with SVMs

Forward each region through ConvNet

Warped image regions (224x224 pixels)

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

Problem: Very slow! Need to do ~2k independent forward passes for each image!

Idea: Pass the image through convnet before cropping! Crop the conv feature instead!

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015source. Reproduced with permission.

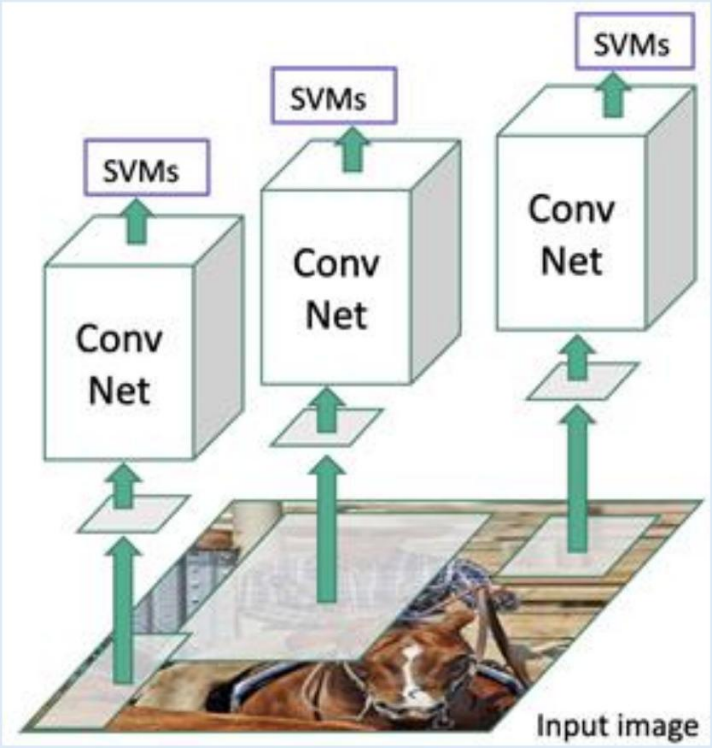
Fast R-CNN



Input image

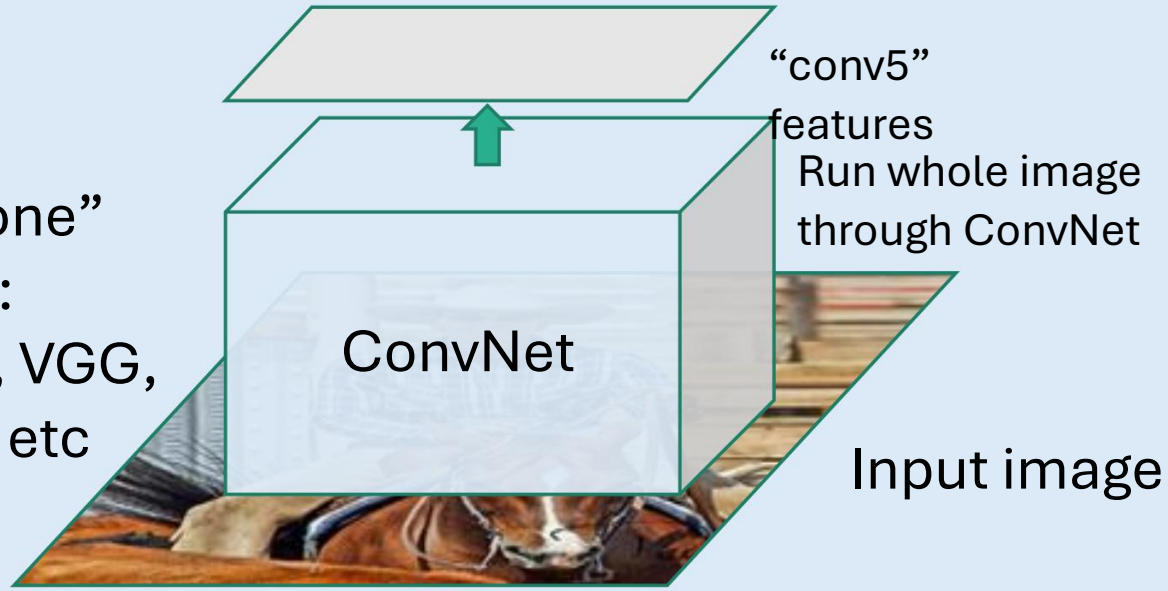
Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015source. Reproduced with permission.

“Slow” R-CNN



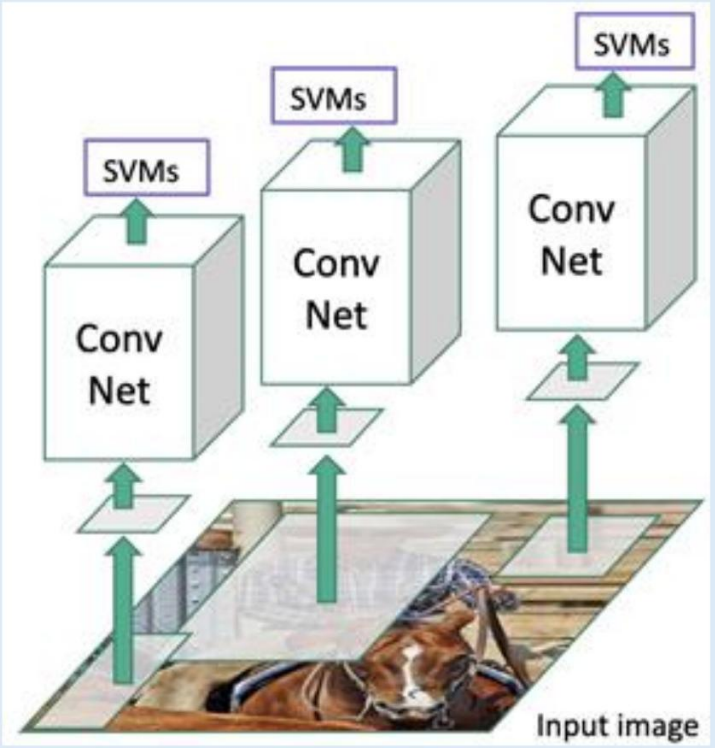
Fast R-CNN

“Backbone” network: AlexNet, VGG, ResNet, etc



Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015source. Reproduced with permission.

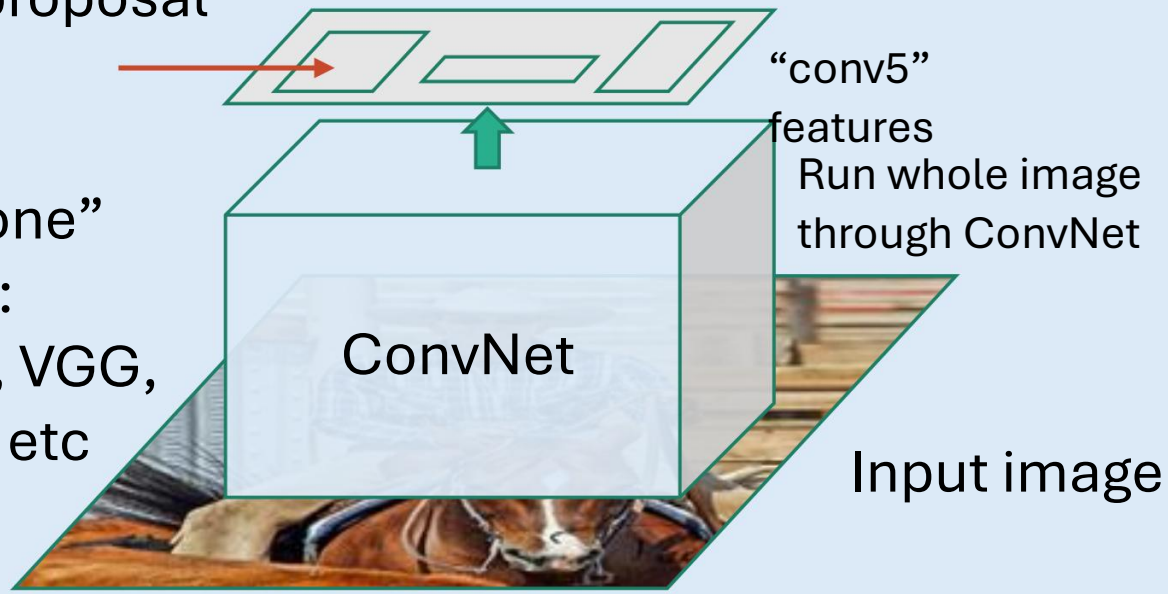
“Slow” R-CNN



Fast R-CNN

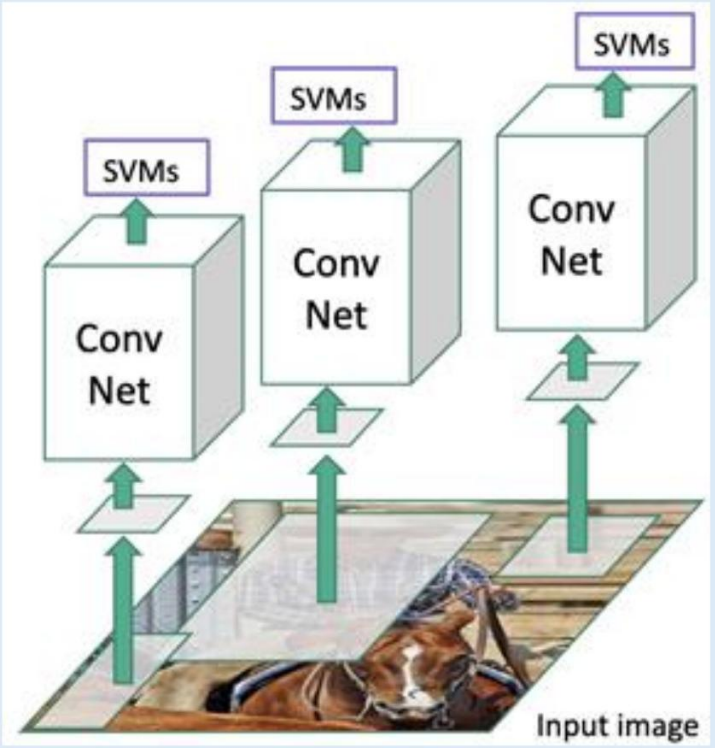
Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015source. Reproduced with permission.

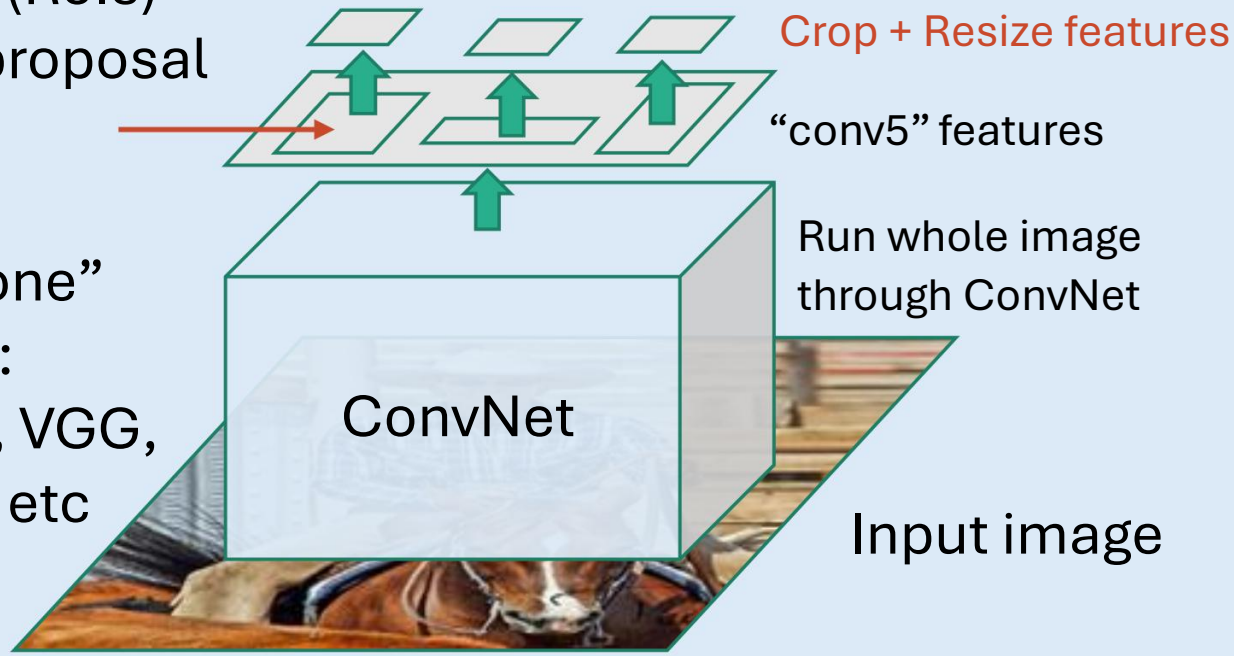
"Slow" R-CNN



Fast R-CNN

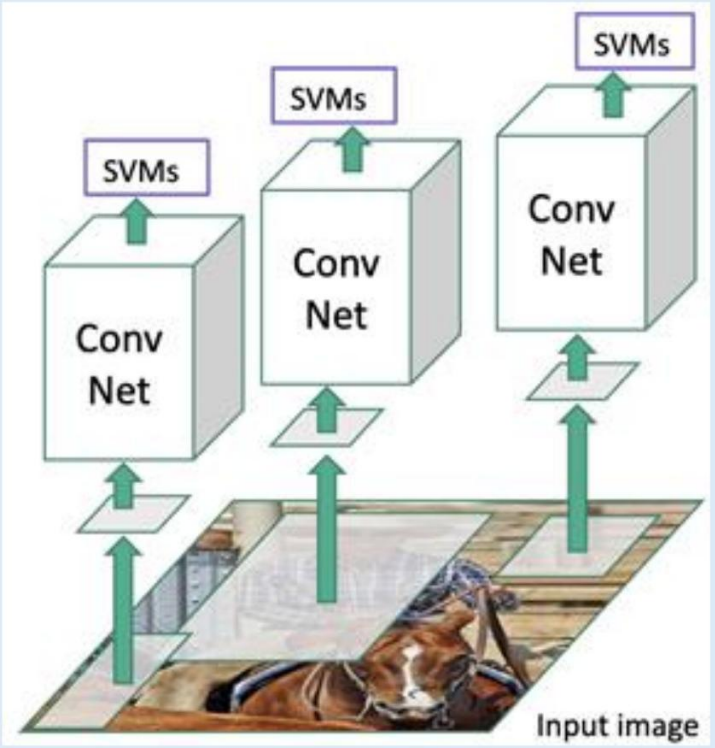
Regions of Interest (RoIs) from a proposal method

"Backbone" network: AlexNet, VGG, ResNet, etc

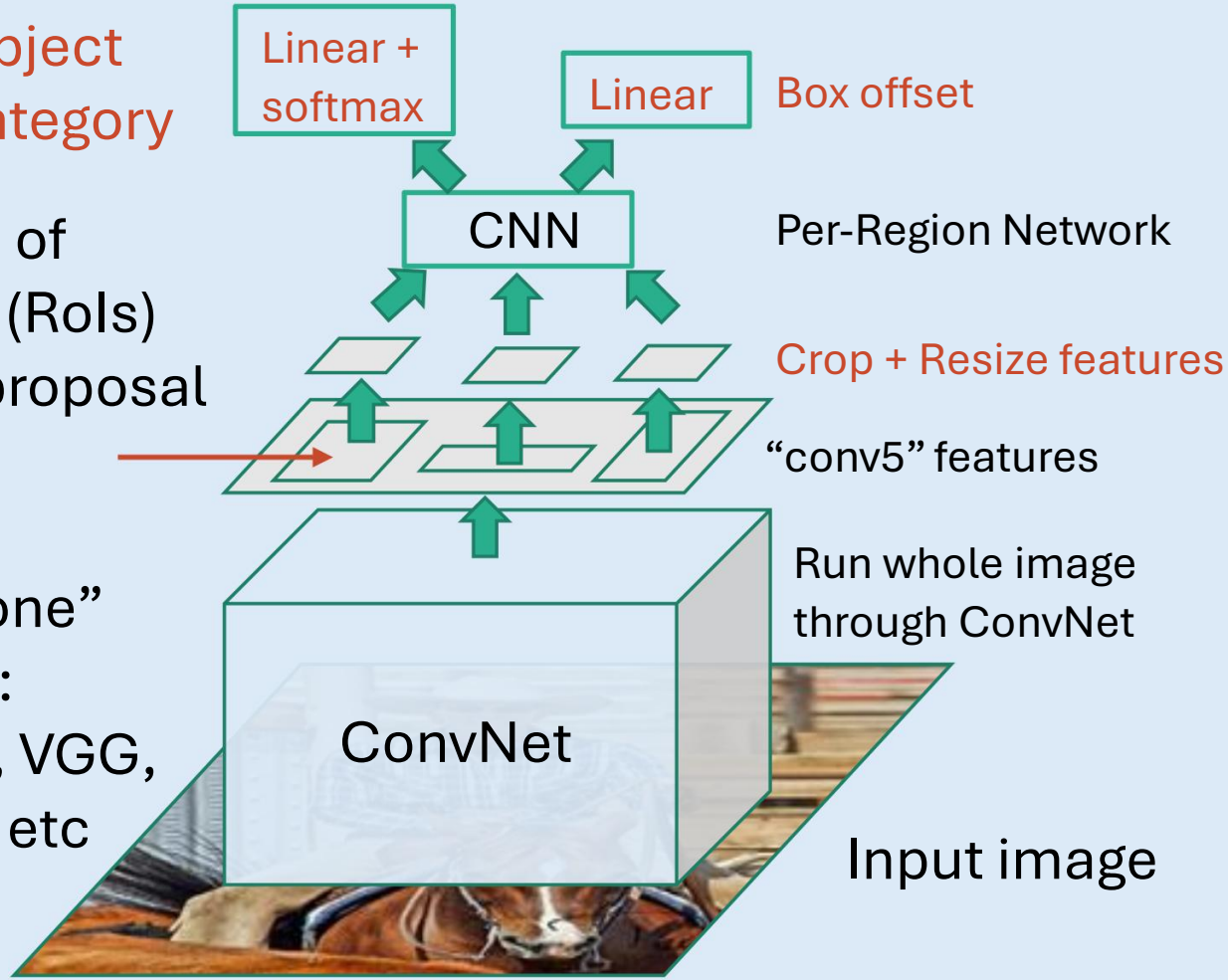


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015source. Reproduced with permission.

"Slow" R-CNN

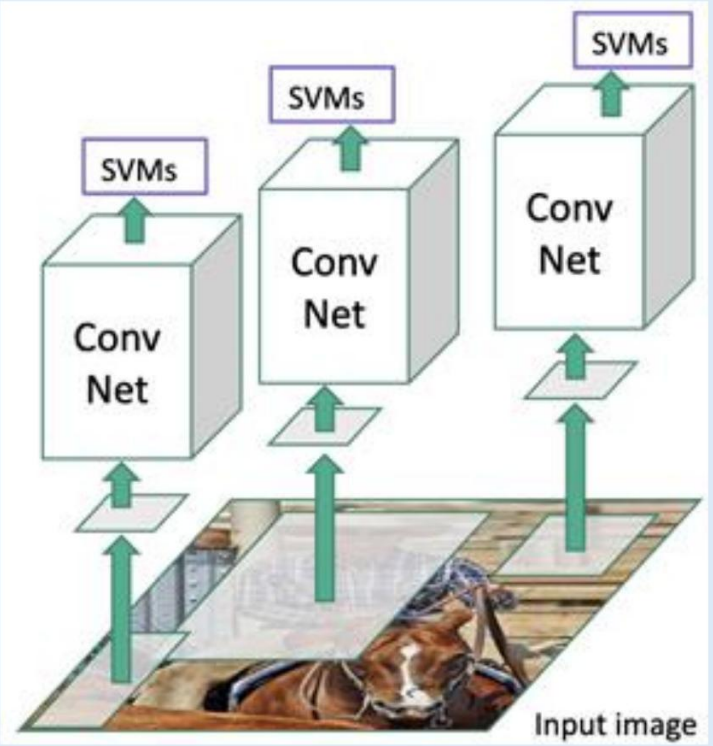


Fast R-CNN

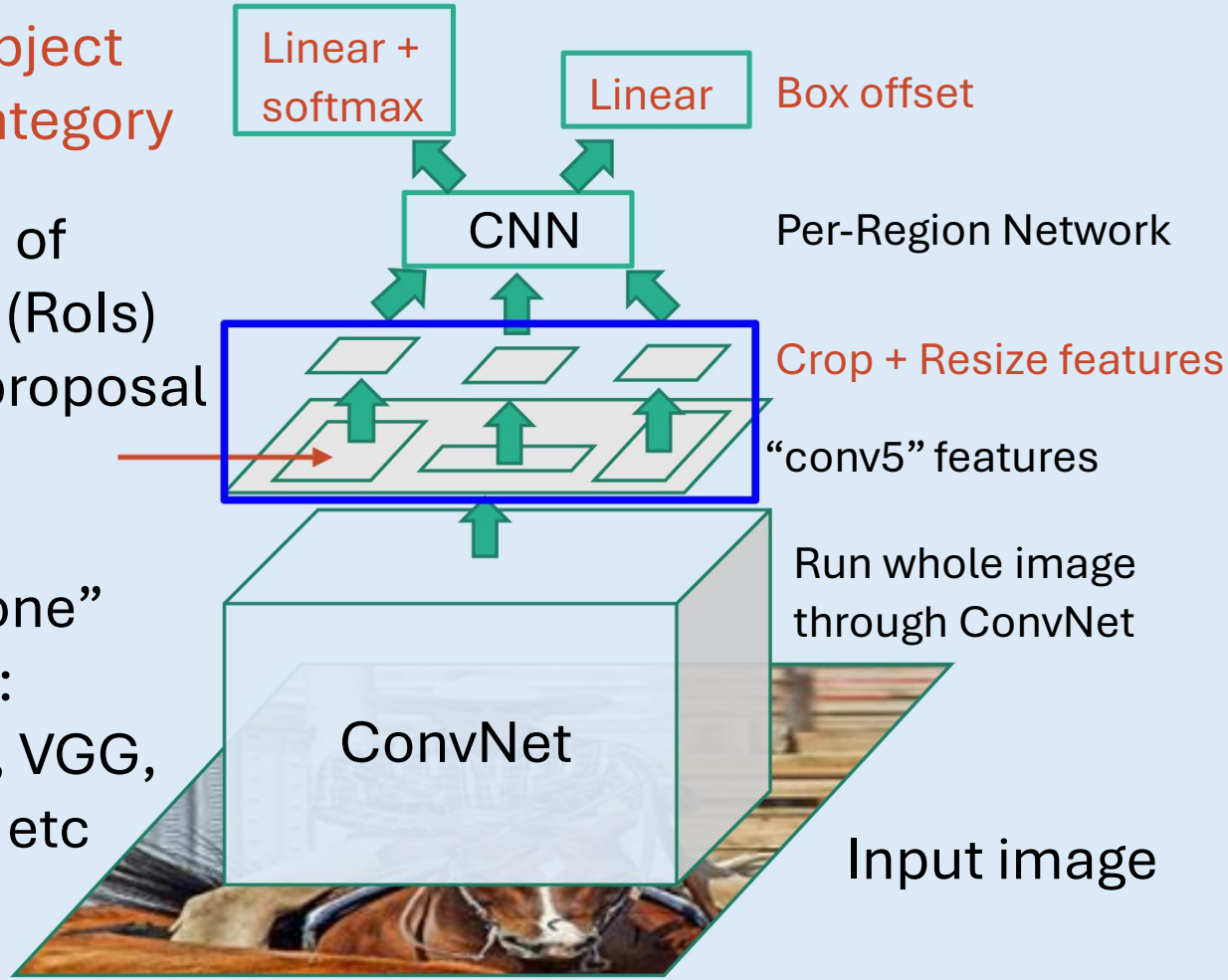


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015source. Reproduced with permission.

"Slow" R-CNN

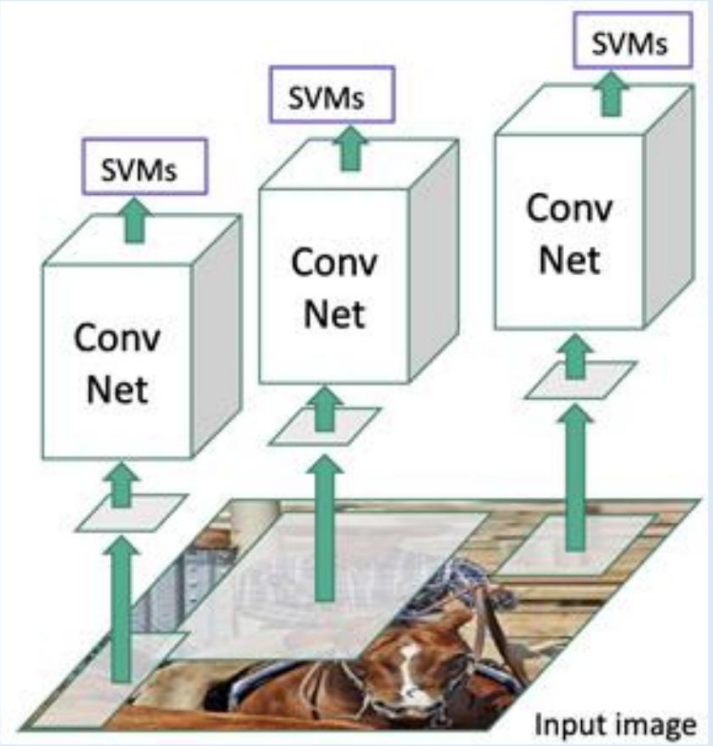


Fast R-CNN

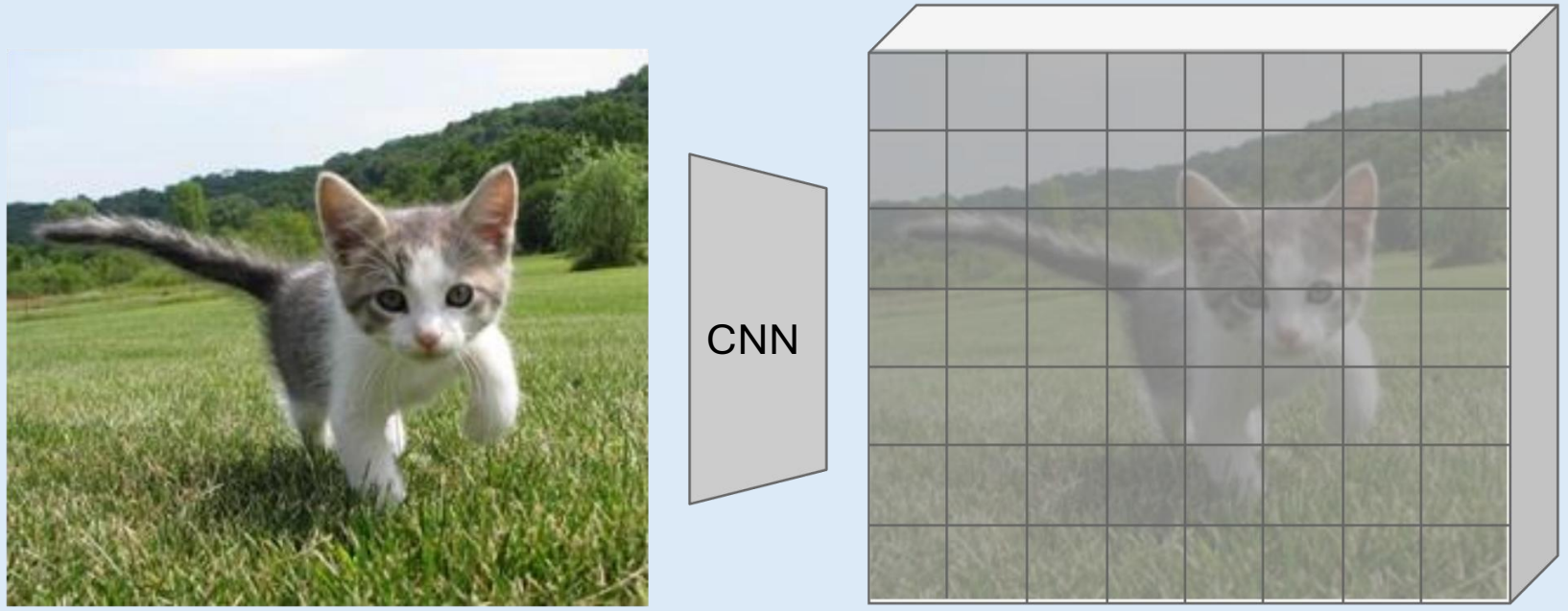


Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015source. Reproduced with permission.

"Slow" R-CNN



Region Proposal Network

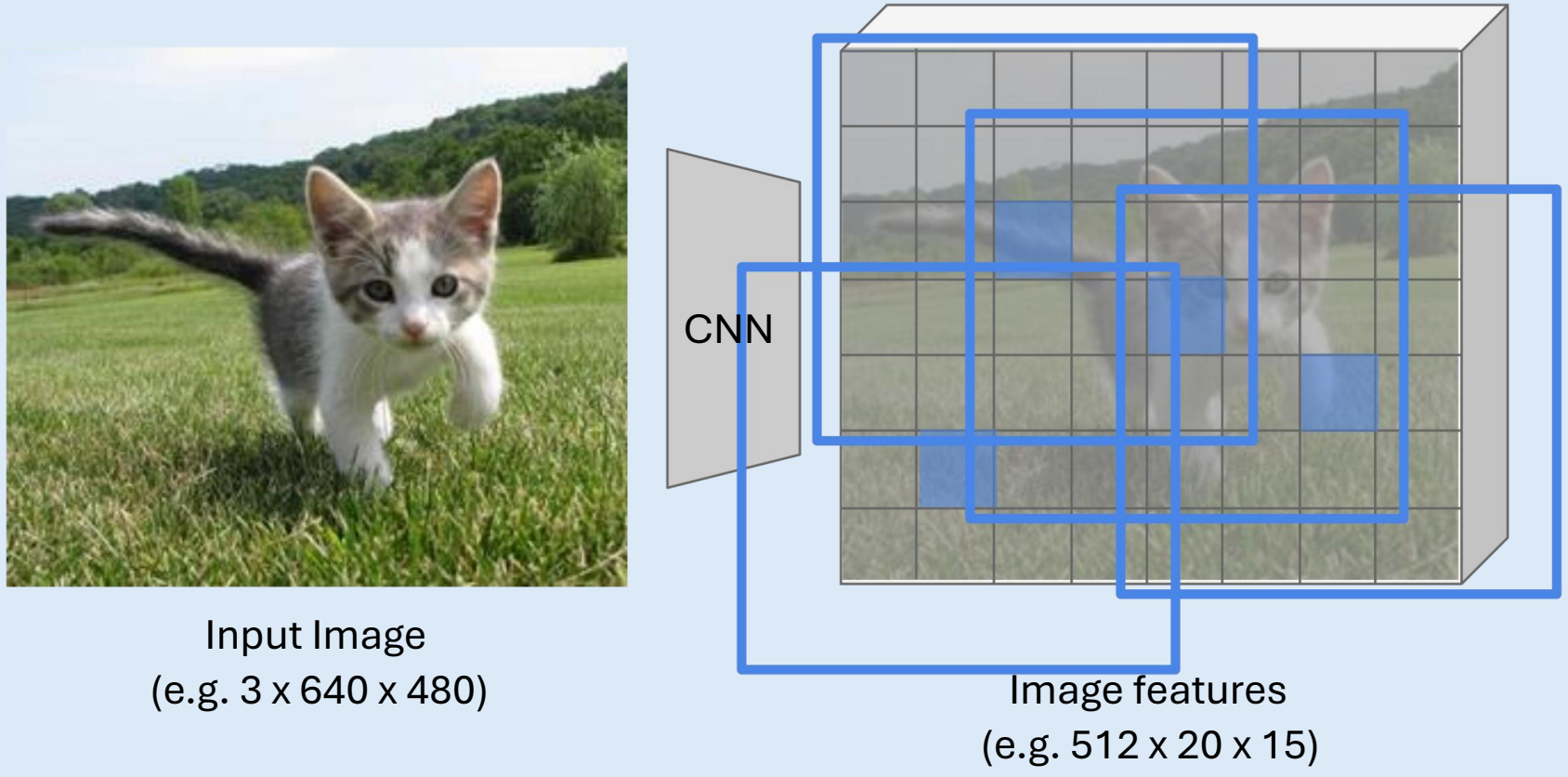


Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map



Input Image
(e.g. 3 x 640 x 480)

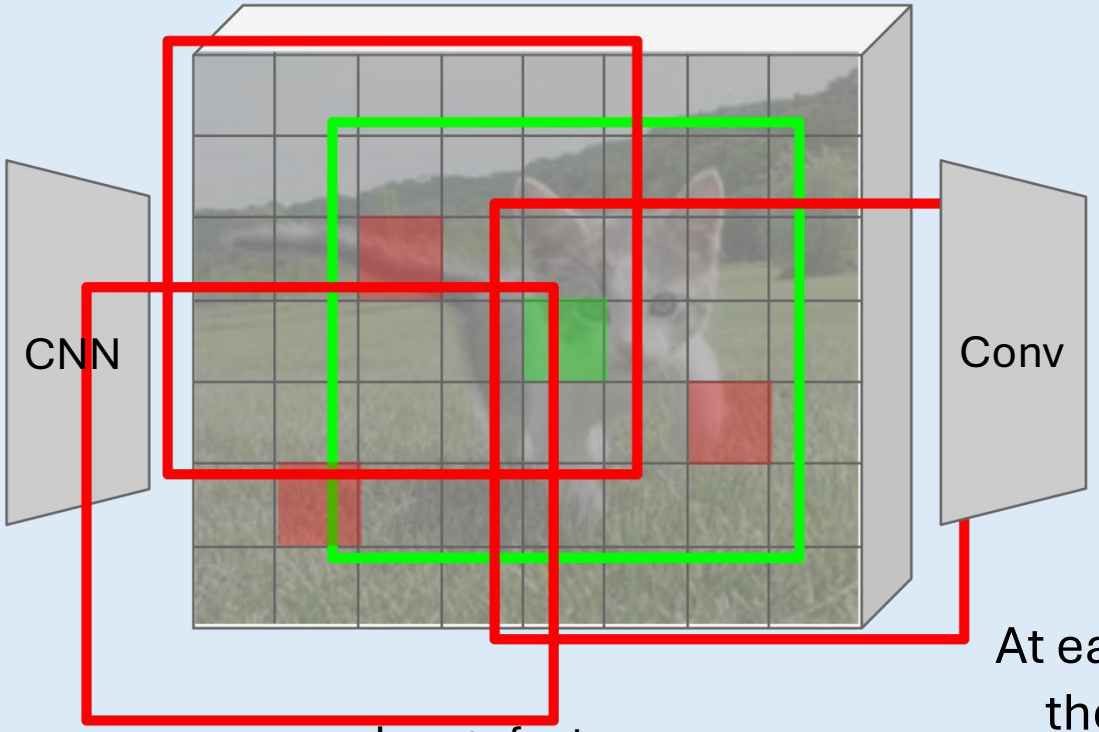


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (binary classification)

Region Proposal Network

Imagine an anchor box of fixed size at each point in the feature map



Input Image
(e.g. 3 x 640 x 480)

CNN

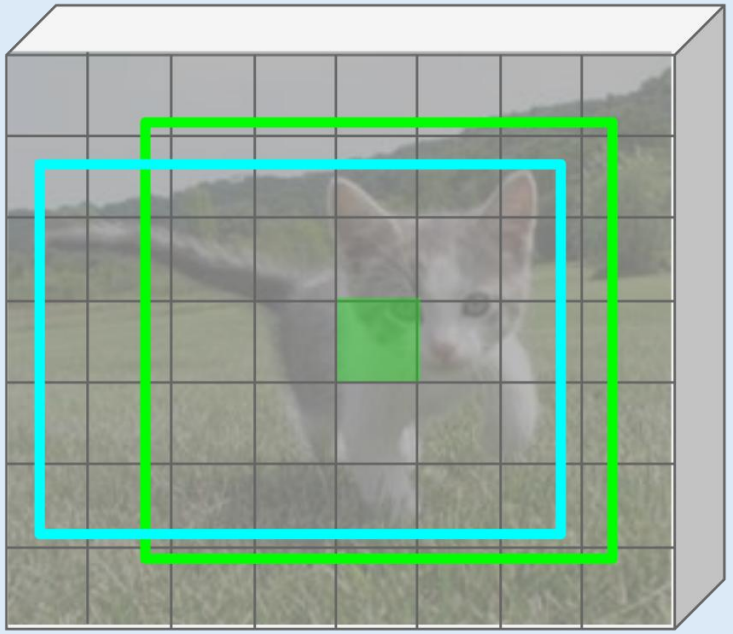


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
1 x 20 x 15

Box corrections
4 x 20 x 15

For positive boxes, also predict a corrections from the anchor to the ground-truth box (regress 4 numbers per pixel)

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point



Input Image
(e.g. 3 x 640 x 480)

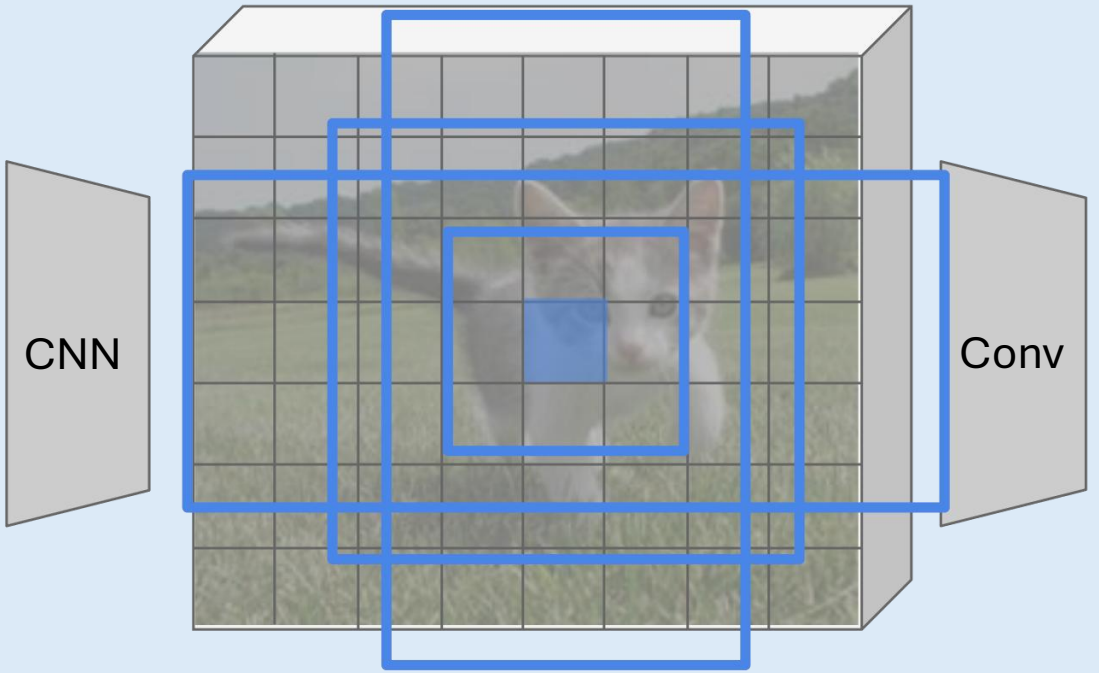


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
K x 20 x 15

Box transforms
4K x 20 x 15

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point



Input Image
(e.g. 3 x 640 x 480)

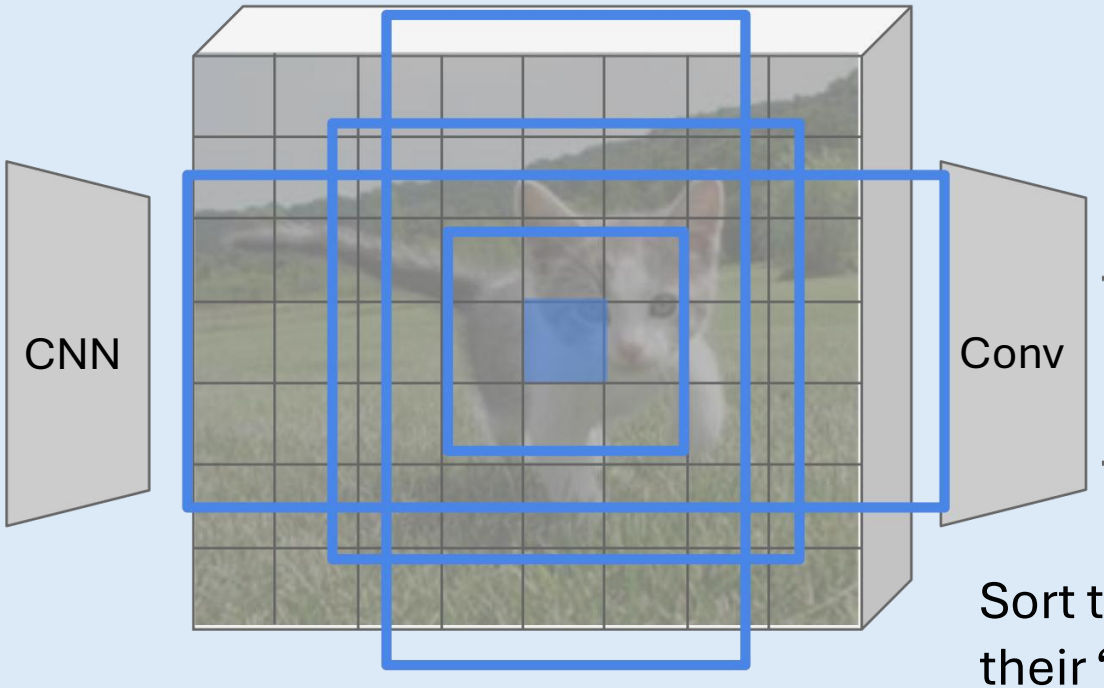


Image features
(e.g. 512 x 20 x 15)

Anchor is an object?
K x 20 x 15

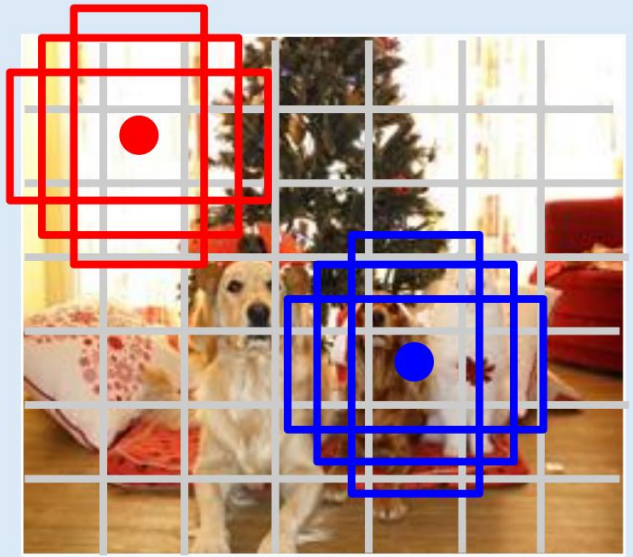
Box transforms
4K x 20 x 15

Sort the K*20*15 boxes by their “objectness” score, take top ~300 as our proposals

Single-Stage Object Detectors: YOLO / SSD / RetinaNet



Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of base boxes centered at each grid cell
Here B = 3

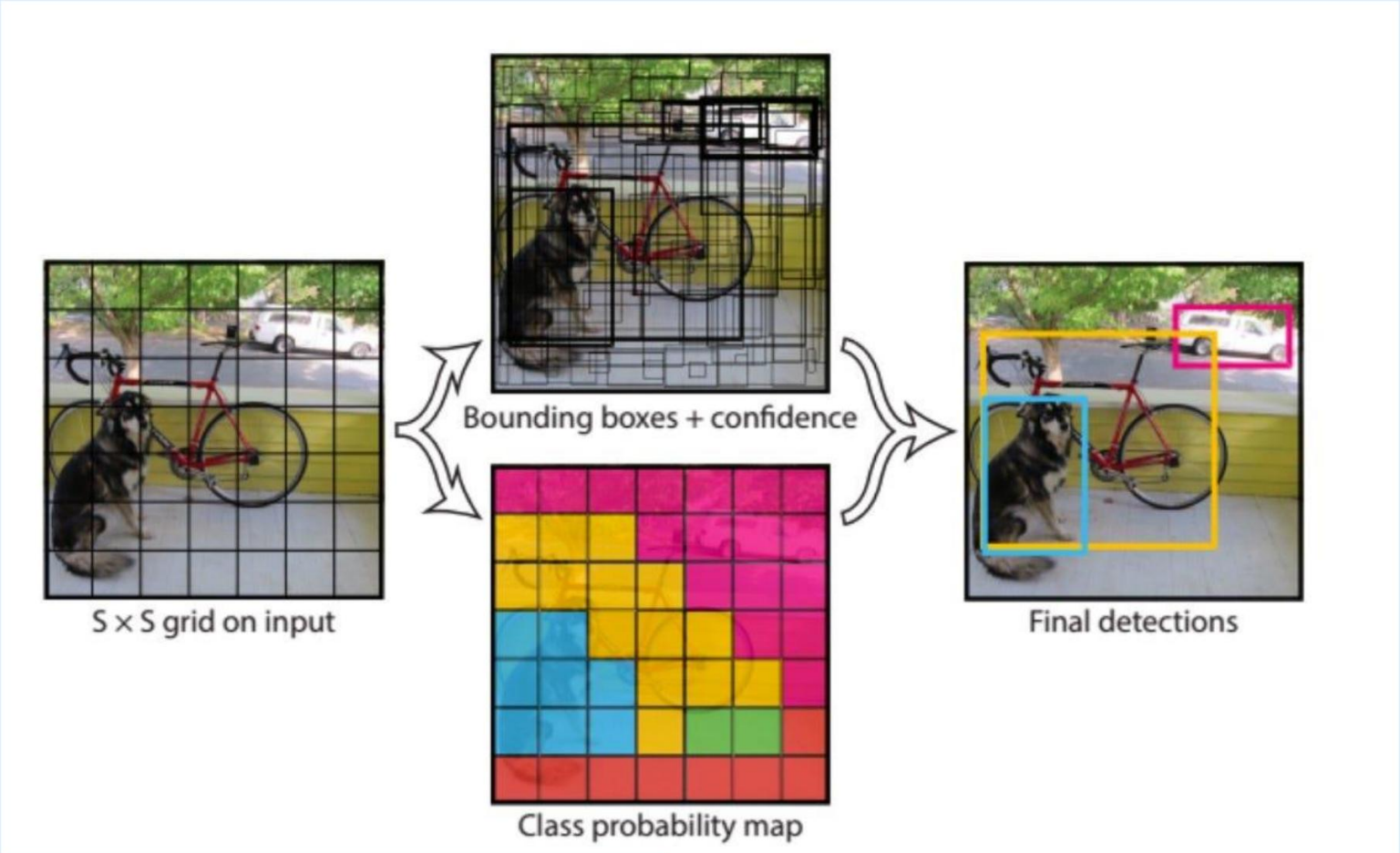
Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)
- Looks a lot like RPN, but category-specific!

Output:
7 x 7 x (5 * B + C)

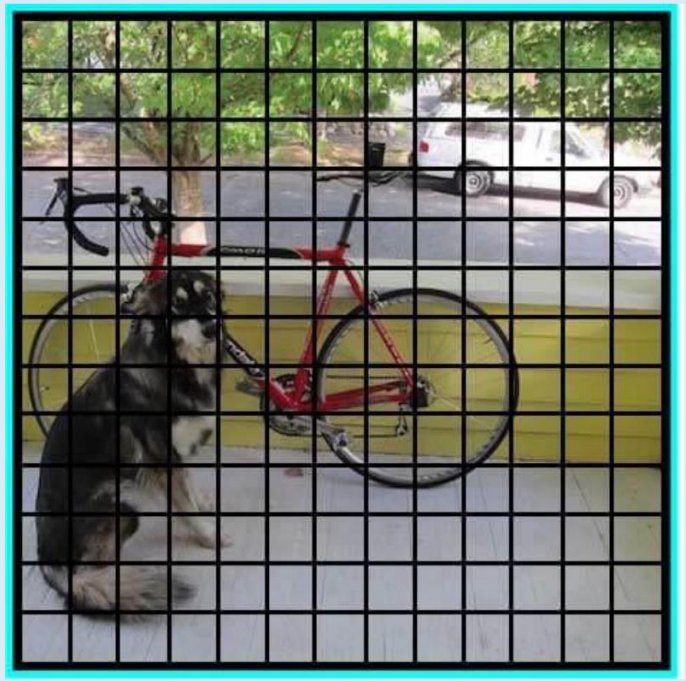
Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

YOLO (You Only Look Once) real-time object detection



Redmon et al. "You only look once: unified, real-time object detection (2015)."

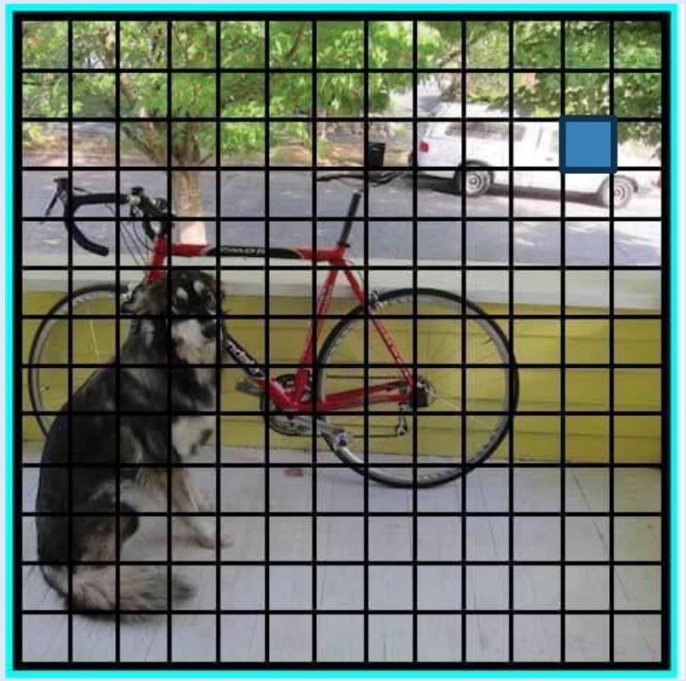
YOLO



SxS Grid

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO



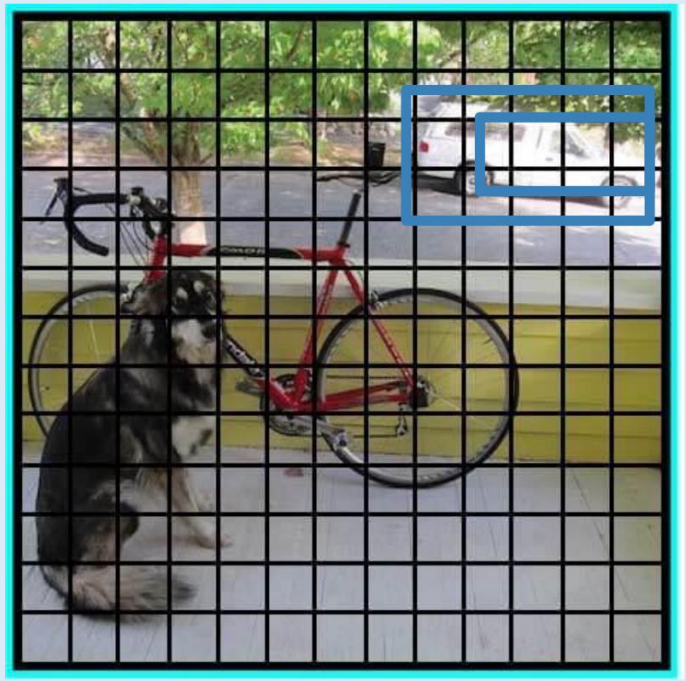
SxS Grid

For each box output:

- P(object): probability that the box contains an object
- B bounding boxes (x, y, h, w)
- P(class): probability of belonging to a class

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO



SxS Grid

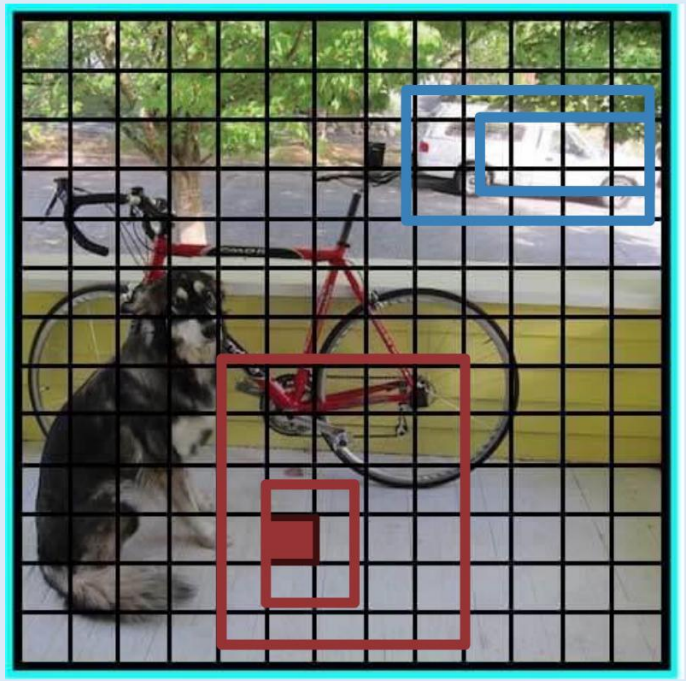
For each box output:

- P(object): probability that the box contains an object
- B bounding boxes (x, y, h, w)
- P(class): probability of belonging to a class

B=2

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO



SxS Grid

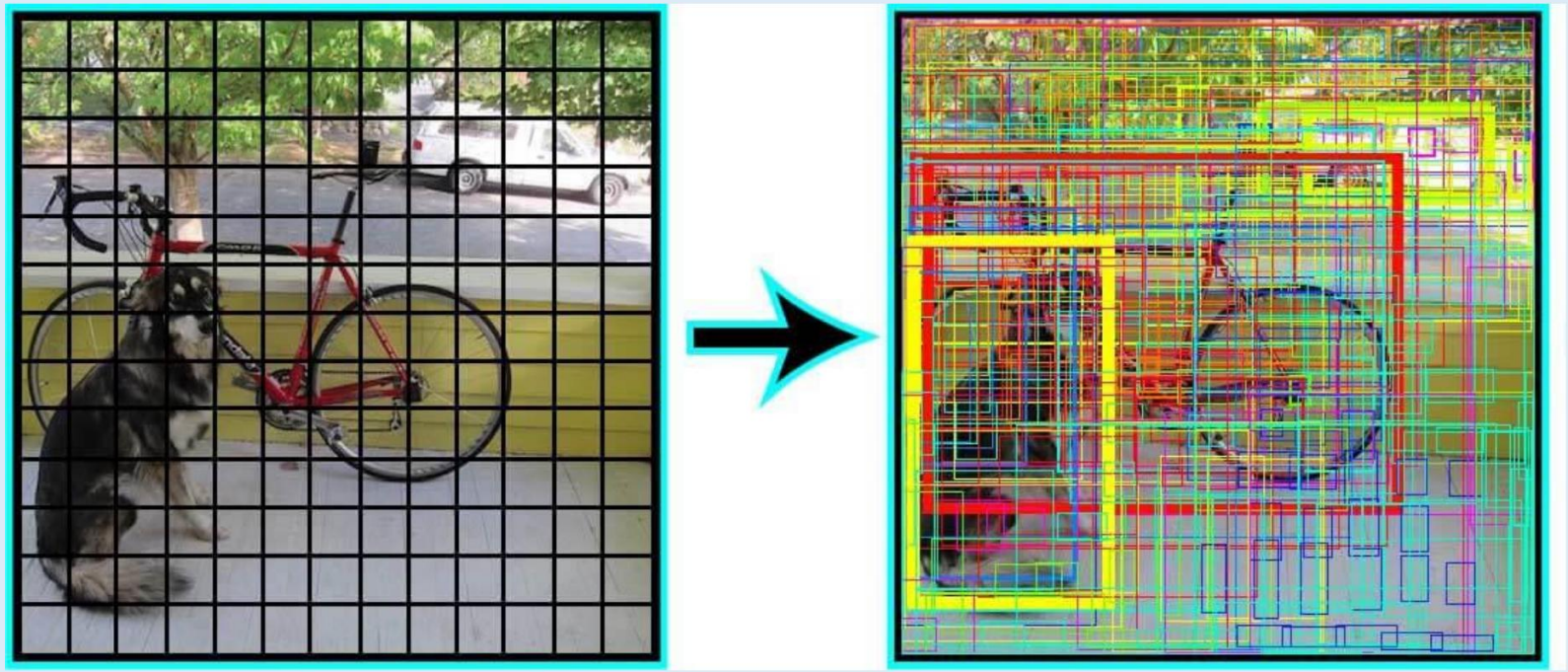
For each box output:

- P(object): probability that the box contains an object
- B bounding boxes (x, y, h, w)
- P(class): probability of belonging to a class

B=2

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO

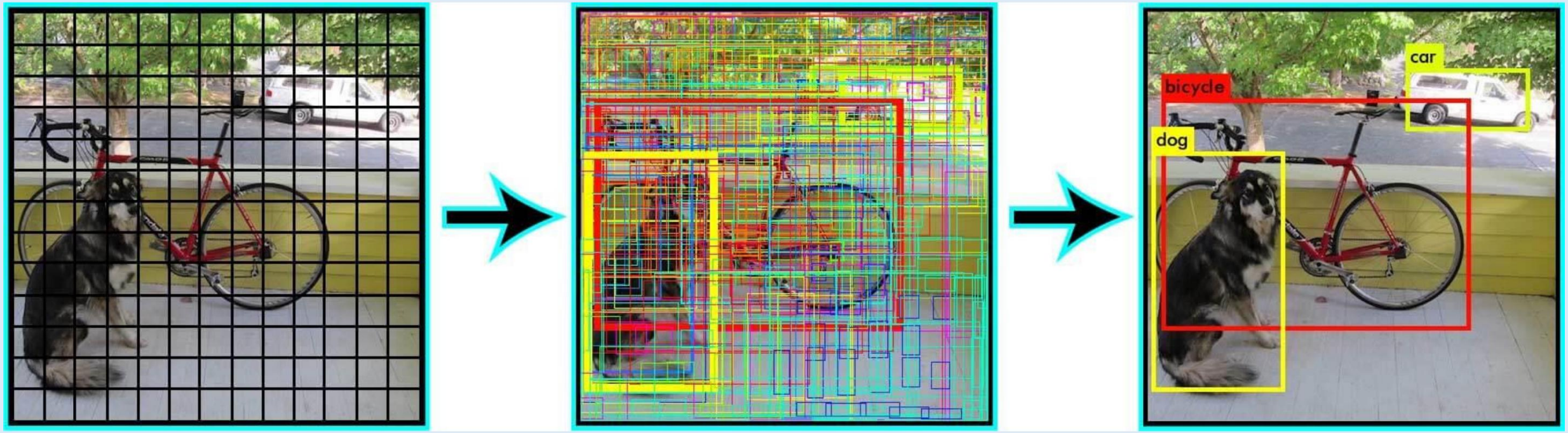


SxS Grid

Many bounding boxes with different object probabilities

Redmon et al. "You only look once: unified, real-time object detection (2015)."

YOLO



SxS Grid

Redmon et al. "You only look once: unified, real-time object detection (2015)."

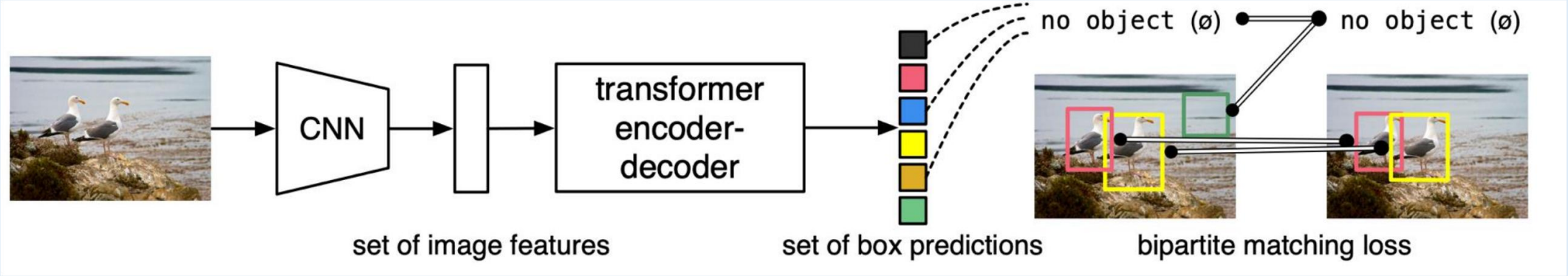
Detection, Segmentation and Understanding

Object **De**tection with **Tr**ansformers: DETR

Simple object detection pipeline: directly output a set of boxes from a Transformer

No anchors, no regression of box transforms

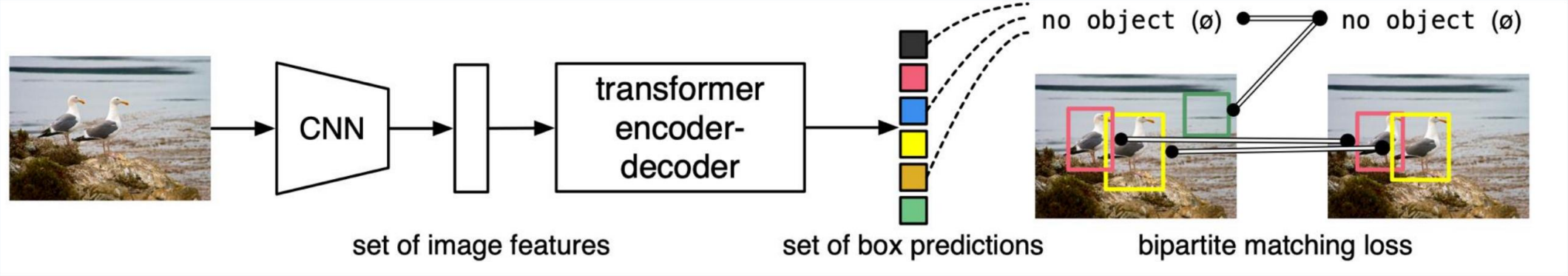
Match predicted boxes to GT boxes with bipartite matching; train to regress box coordinates

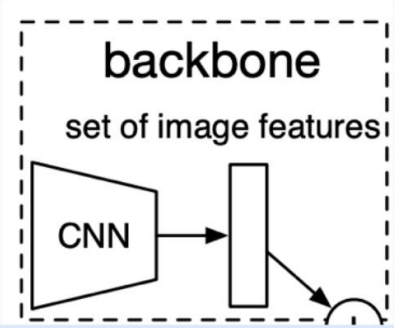


Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

Detection, Segmentation and Understanding

Object Detection with Transformers: DETR

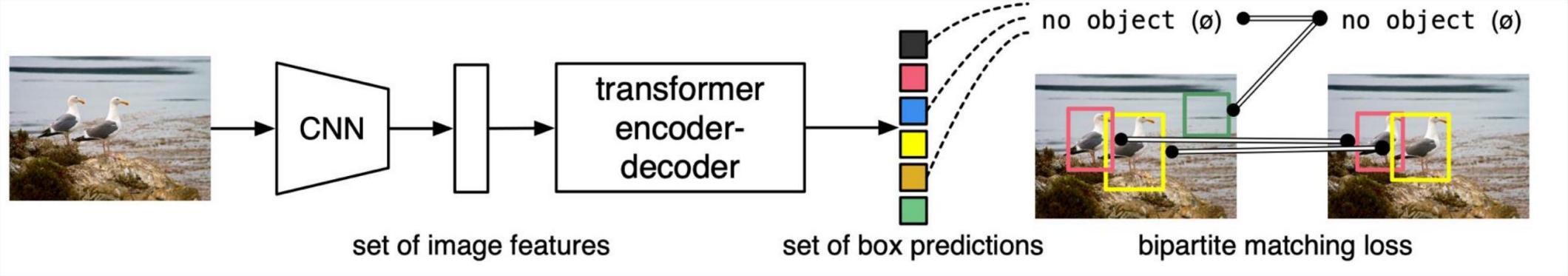


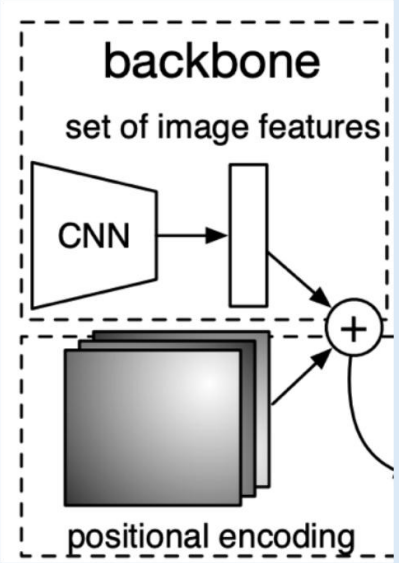


Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

Detection, Segmentation and Understanding

Object Detection with Transformers: DETR

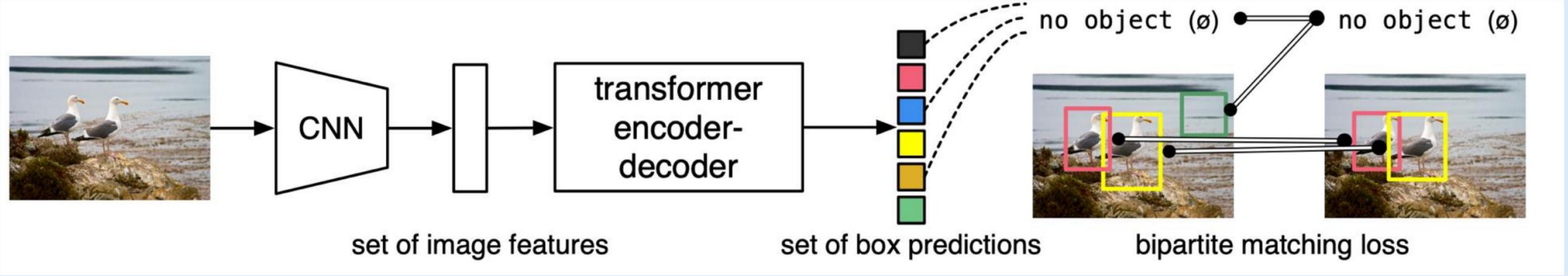


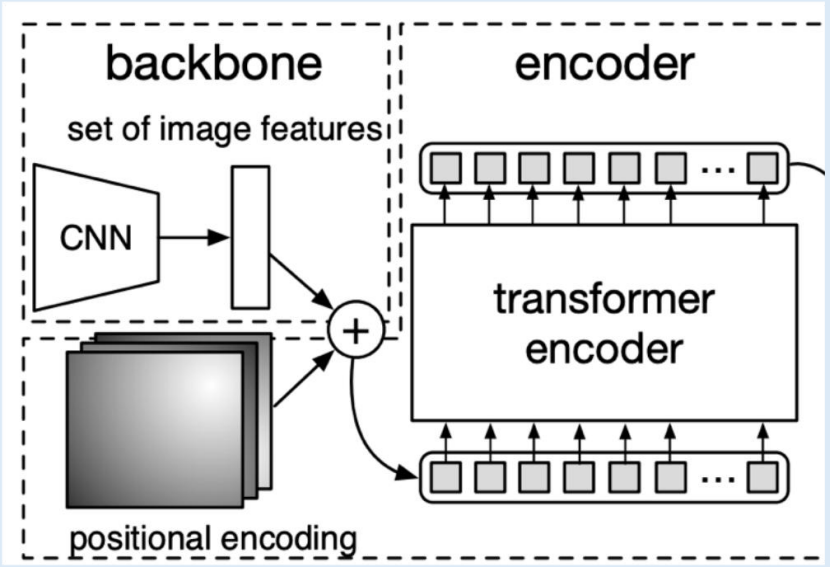


Carion et al, “End-to-End Object Detection with Transformers”, ECCV 2020

Detection, Segmentation and Understanding

Object Detection with Transformers: DETR

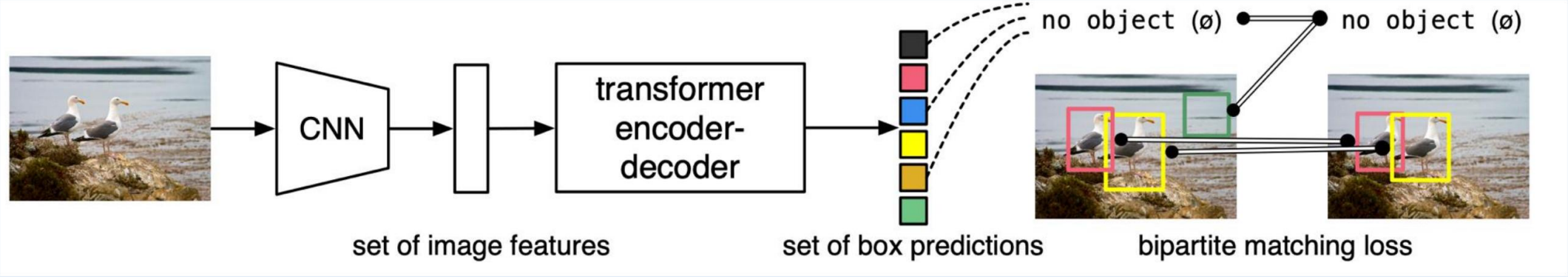


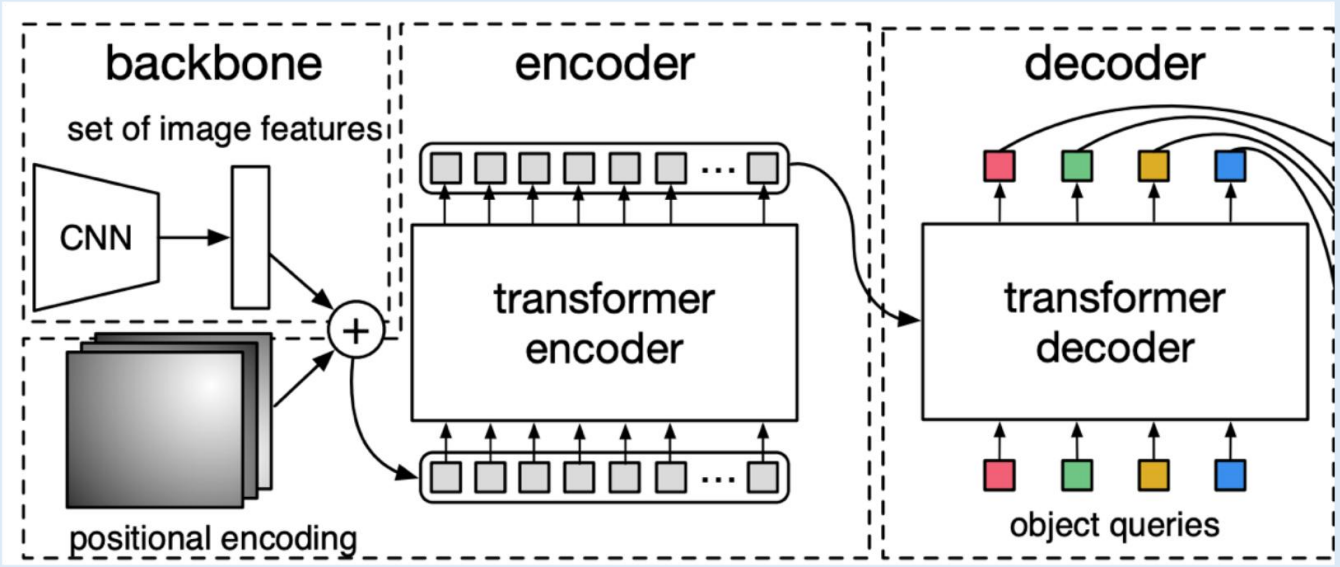


Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Detection, Segmentation and Understanding

Object Detection with Transformers: DETR

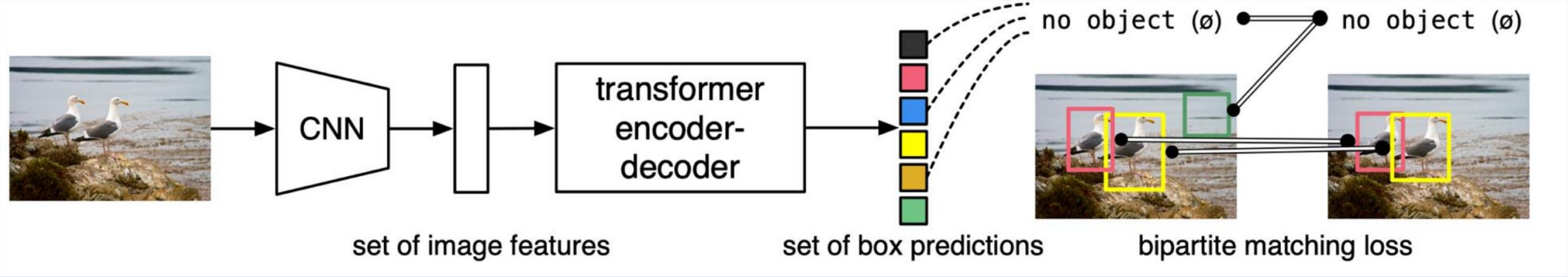


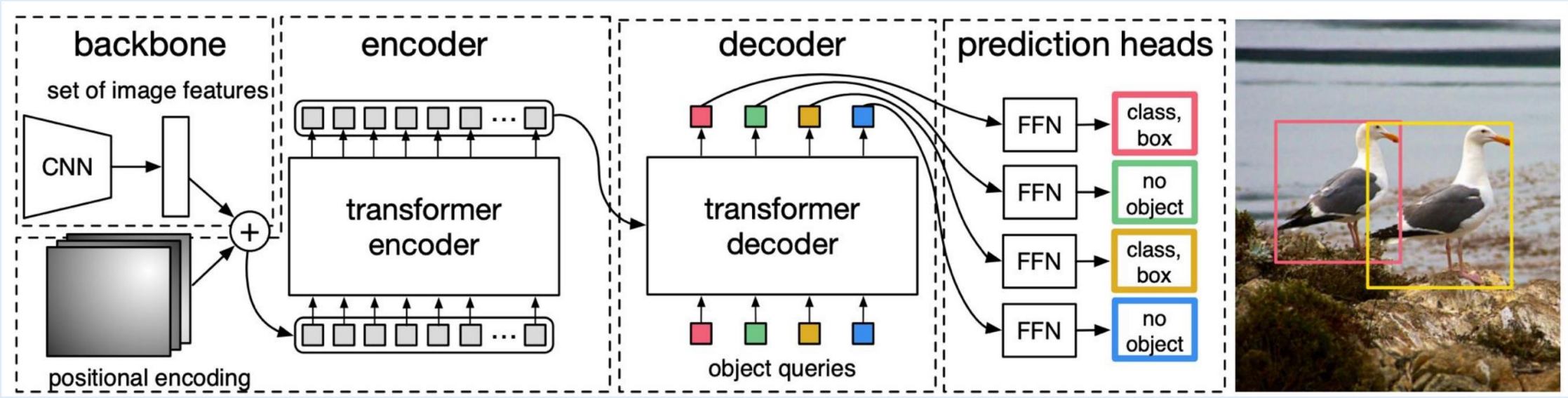


Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Detection, Segmentation and Understanding

Object Detection with Transformers: DETR



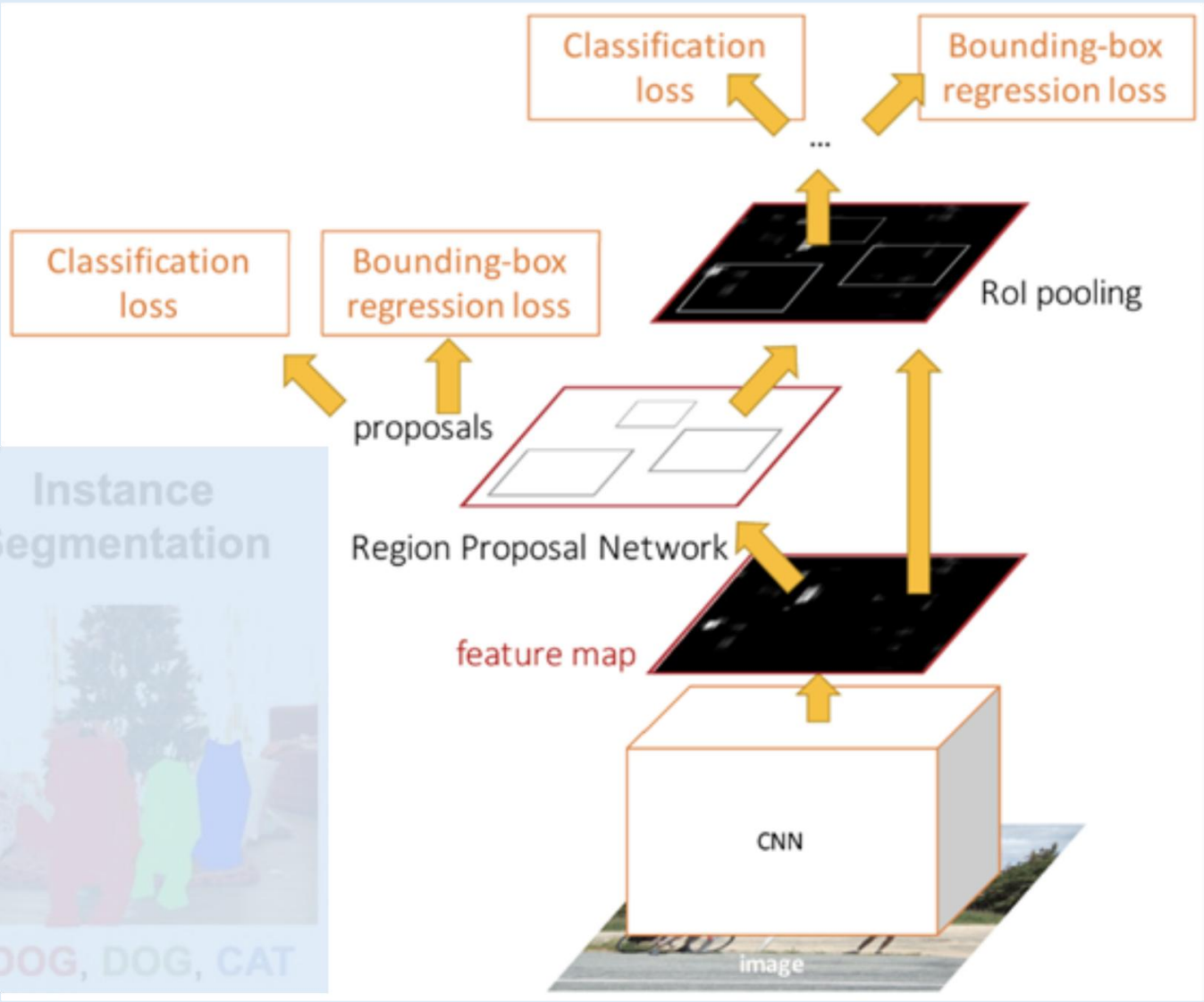


Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Instance Segmentation

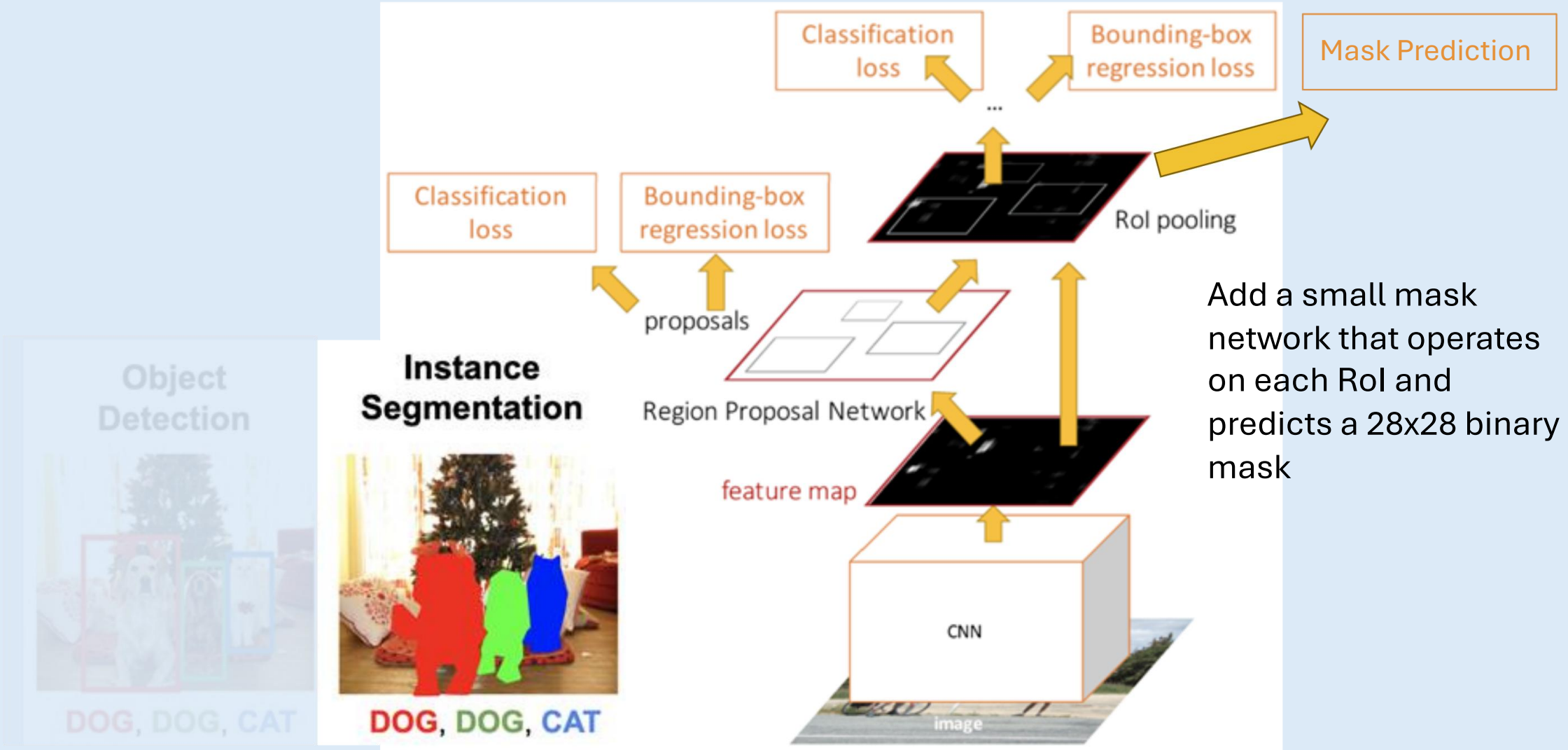
Classification	Semantic Segmentation	Object Detection	Instance Segmentation
			
CAT	GRASS, CAT, TREE, SKY	DOG, DOG, CAT	DOG, DOG, CAT
No spatial extent	No objects, just pixels	Multiple Object	

Object Detection: Faster R-CNN





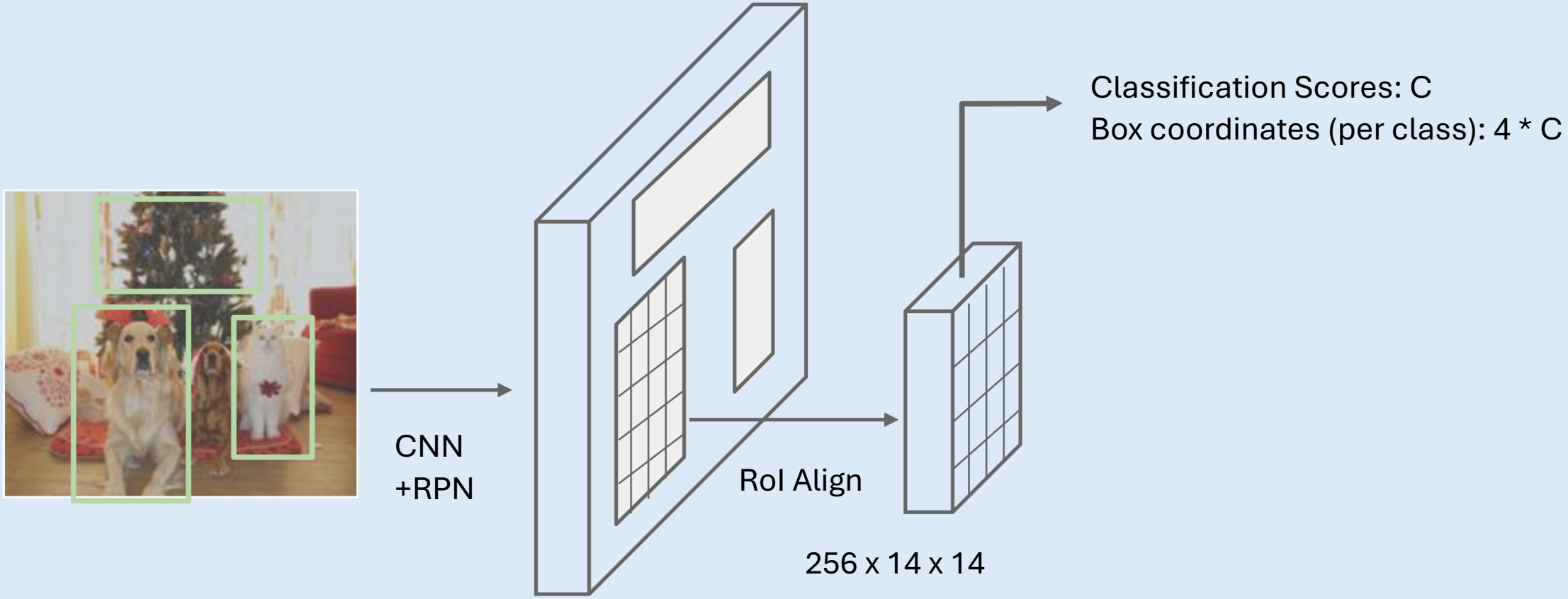
Instance Segmentation: Mask R-CNN



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

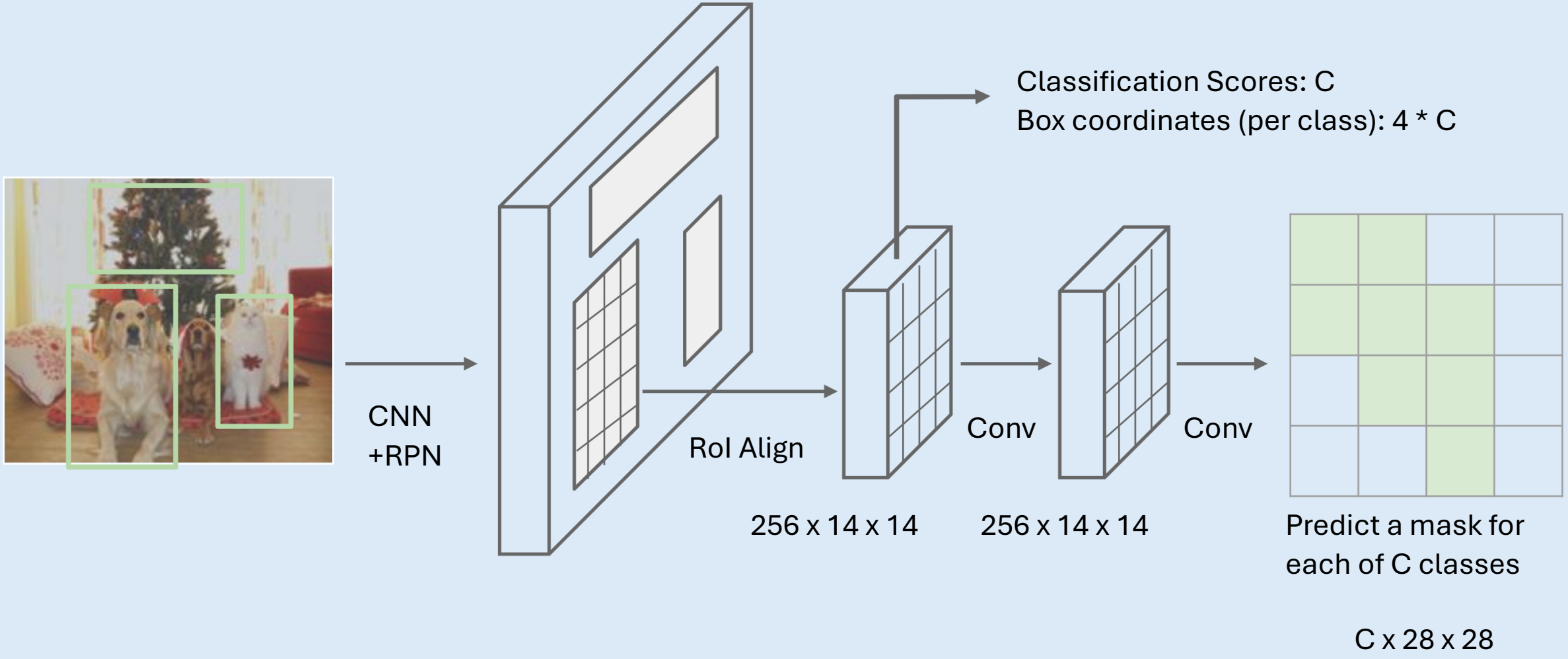
He et al, "Mask R-CNN", ICCV 2017

Mask R-CNN



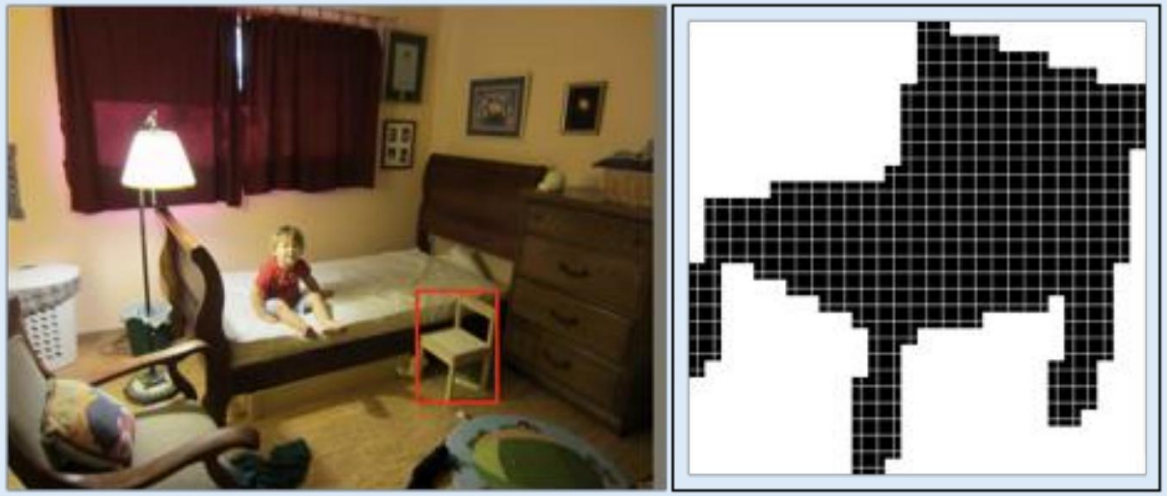
He et al, “Mask R-CNN”, arXiv 2017

Mask R-CNN

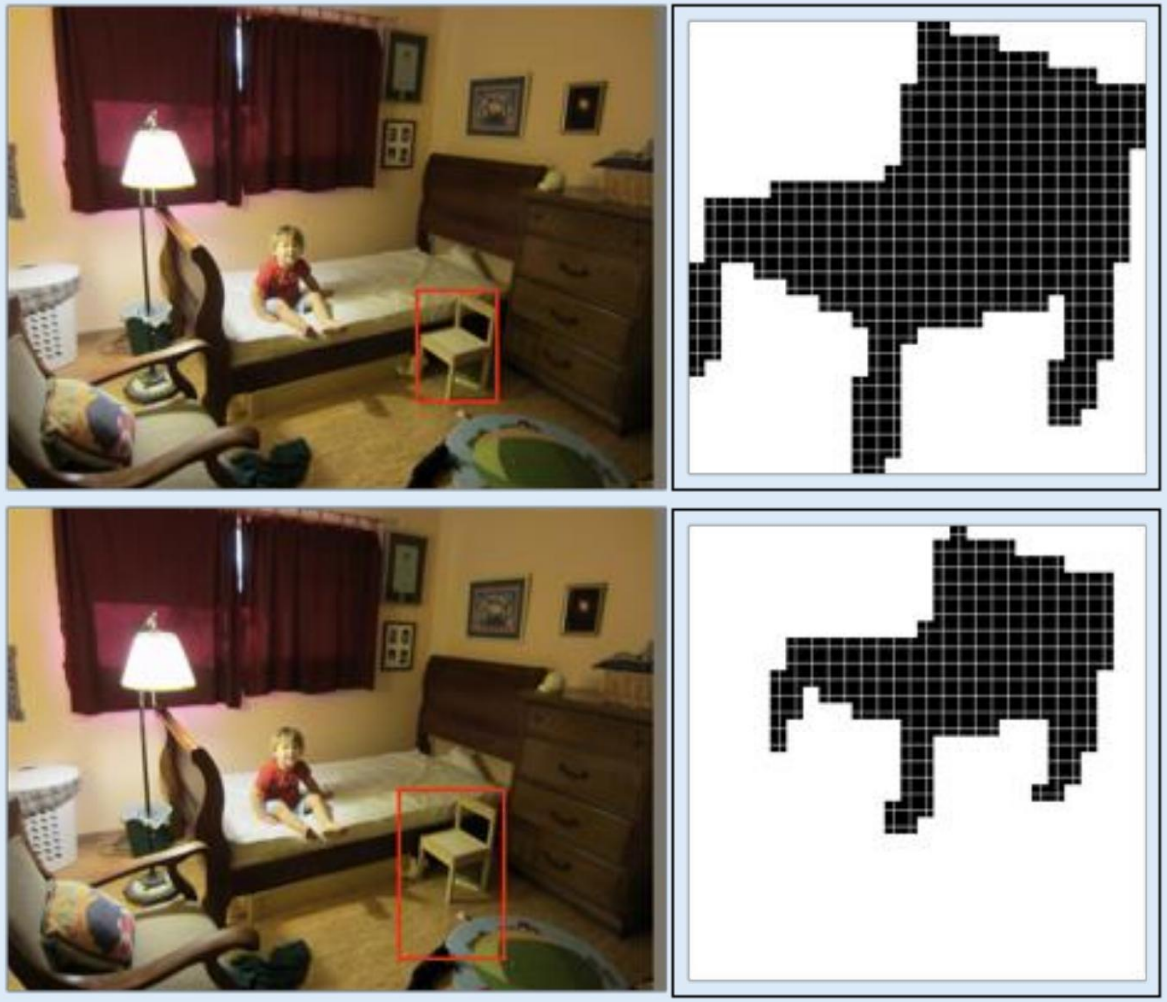


He et al, “Mask R-CNN”, arXiv 2017

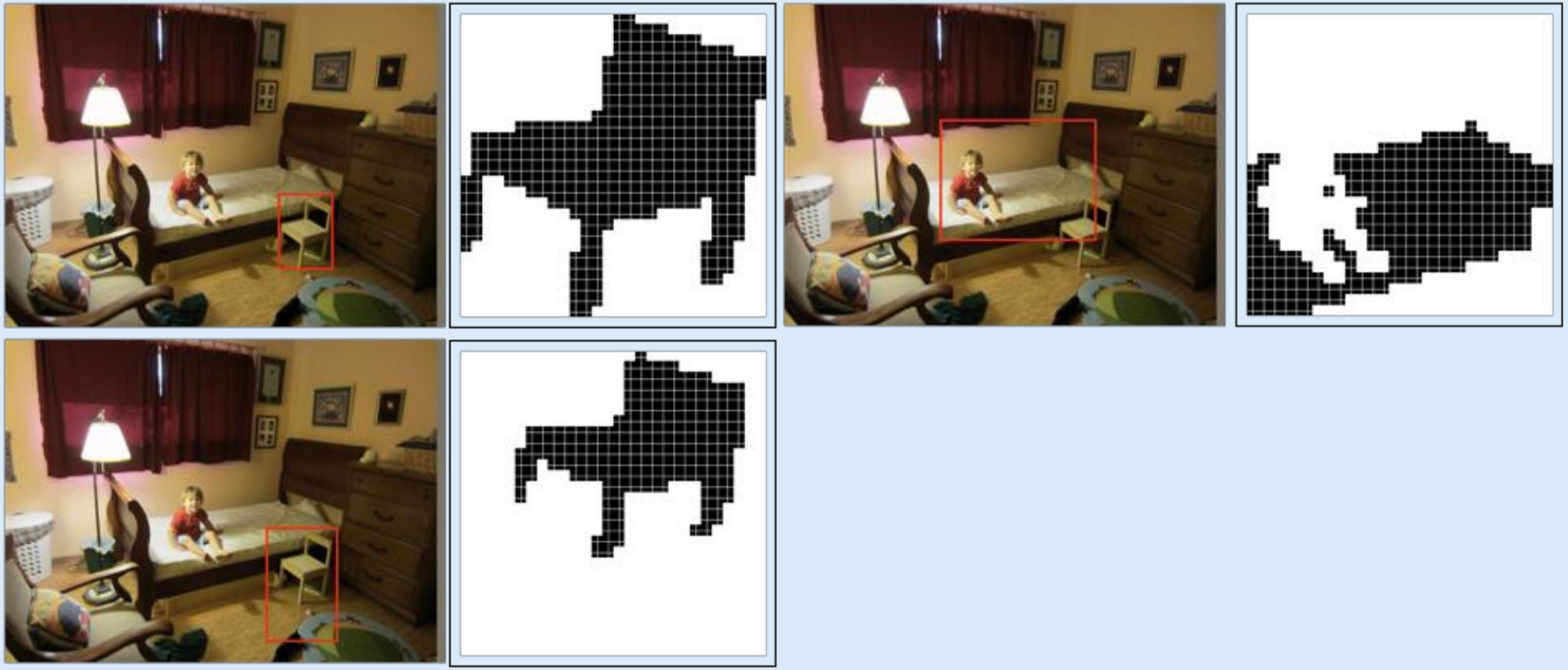
Mask R-CNN: Example Mask Training Targets



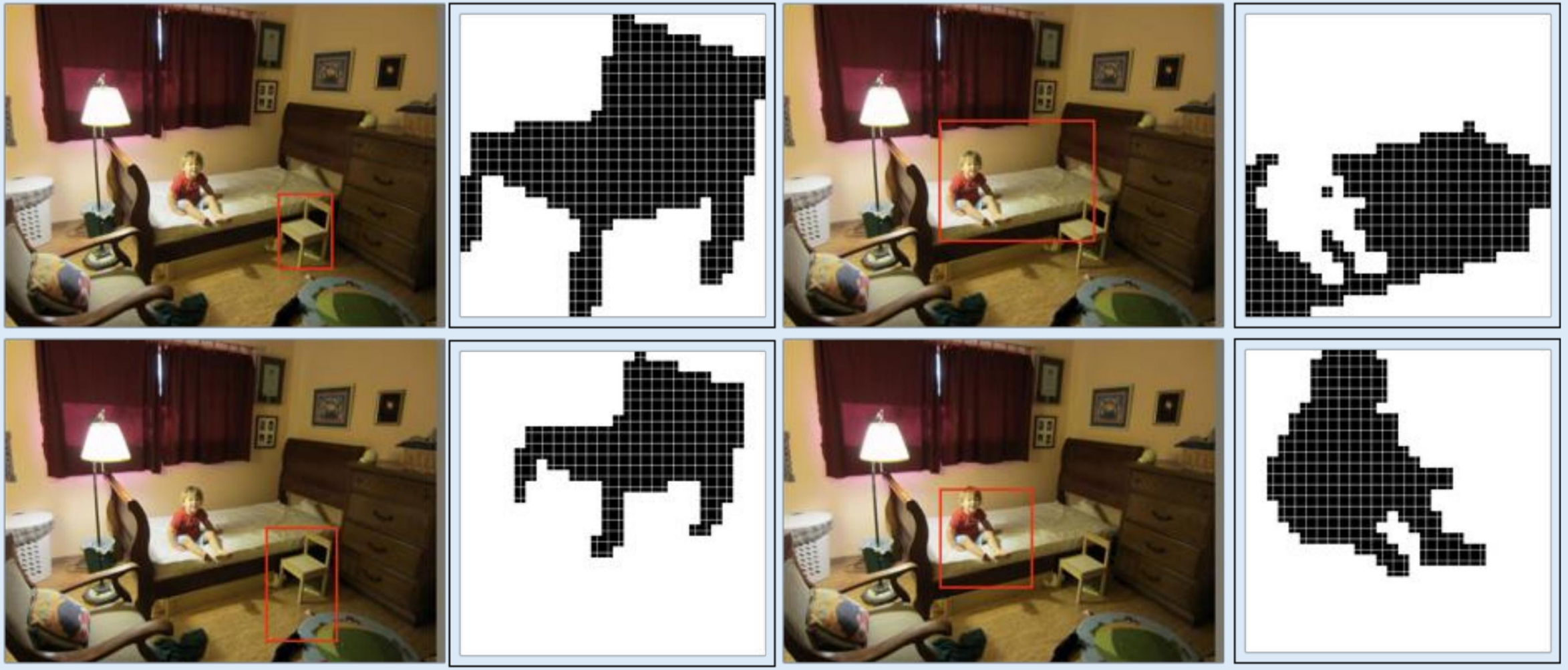
Mask R-CNN: Example Mask Training Targets



Mask R-CNN: Example Mask Training Targets

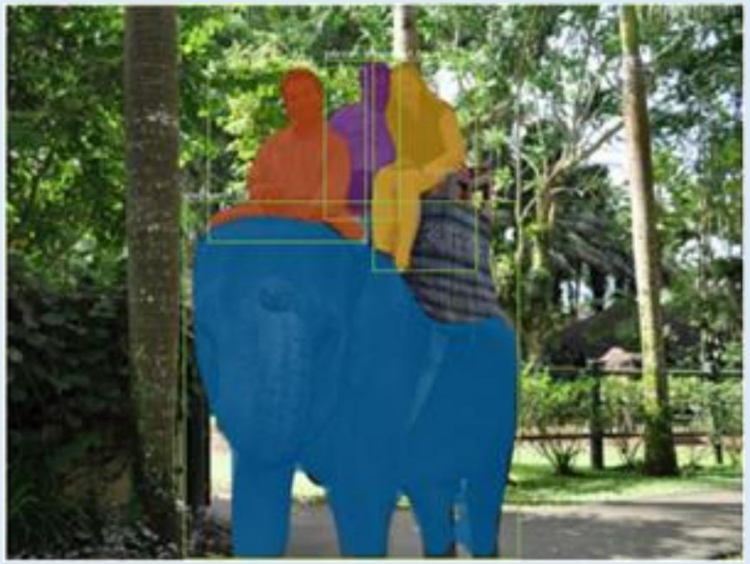


Mask R-CNN: Example Mask Training Targets



Mask R-CNN: Very Good Results!







He et al, “Mask R-CNN”, ICCV 2017

Mask R-CNN Also does pose



He et al, “Mask R-CNN”, ICCV 2017

Recap: Lots of computer vision tasks!

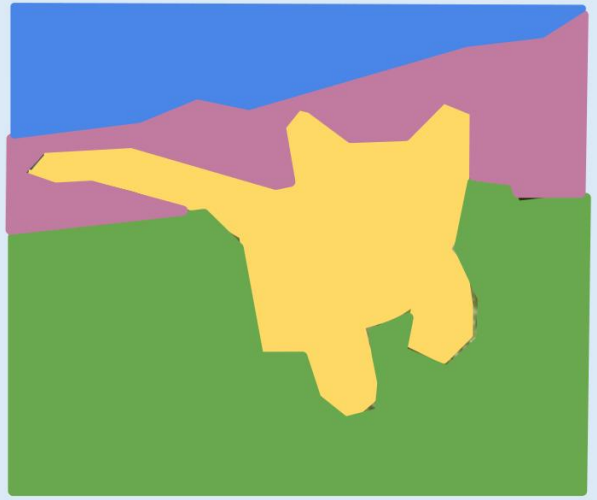
Classification



CAT

No spatial extent

Semantic Segmentation



GRASS, CAT, TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Object

This image is CC0 public domain

Today

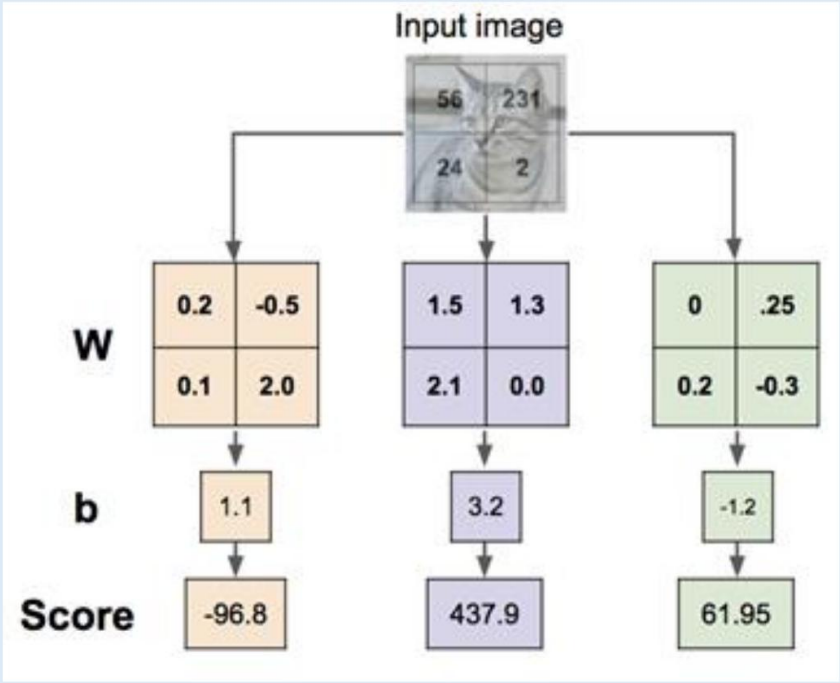
- Transformers Recap

- **Computer Vision Tasks**
  - Semantic Segmentation
  - Object Detection
  - Instance Segmentation

- Visualization & Understanding
  - Model Layers Visualization
  - Saliency Maps
  - CAM & Grad-CAM

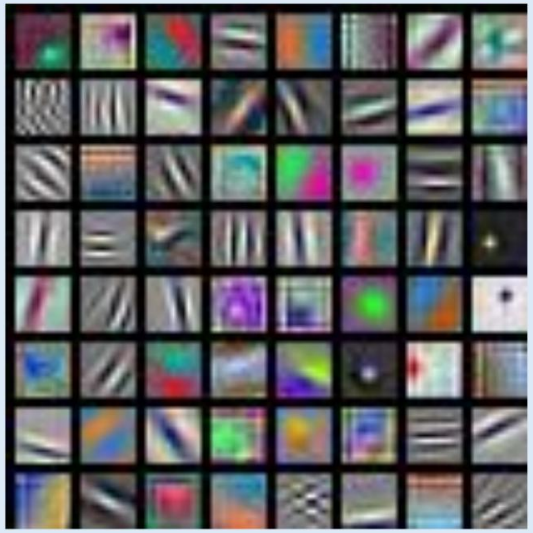
Interpreting a Linear Classifier: Visual Viewpoint



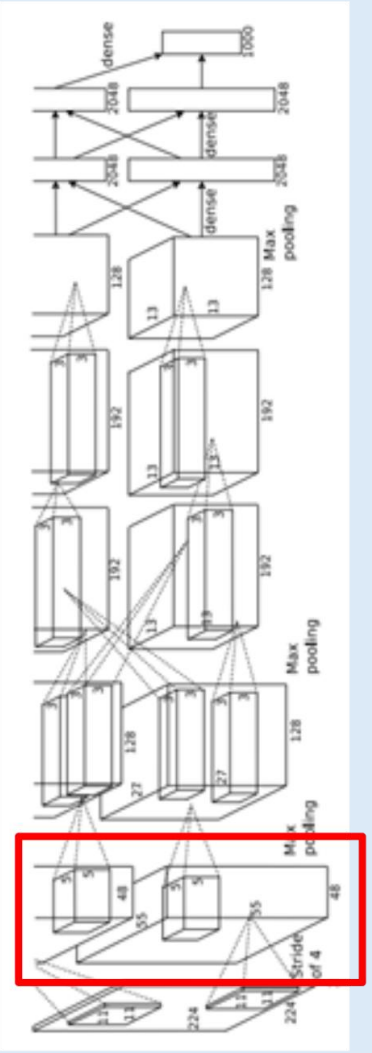




First Layer: Visualize Filters

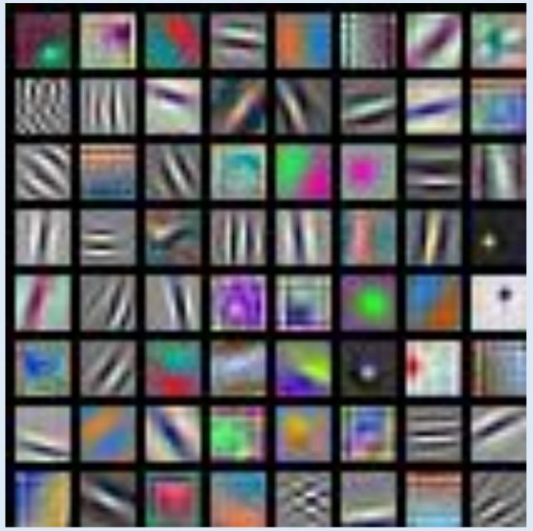


AlexNet:
64 x 3 x 11 x 11



Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

First Layer: Visualize Filters



AlexNet:
64 x 3 x 11 x 11



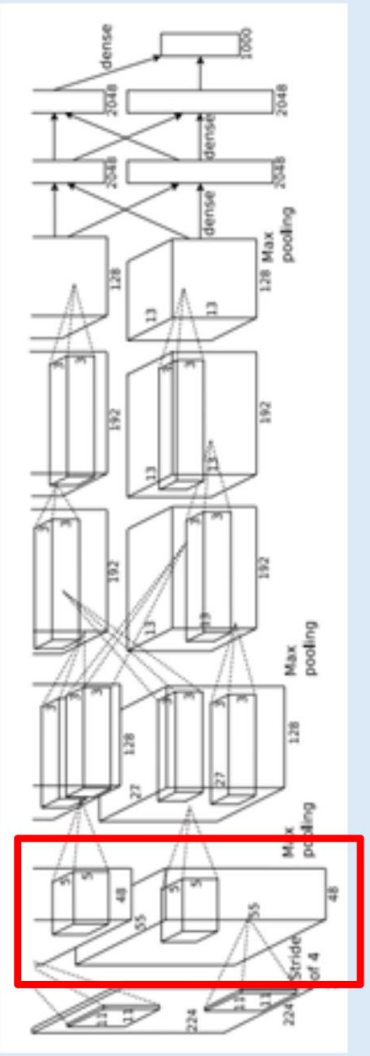
ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7



DenseNet-121:
64 x 3 x 7 x 7

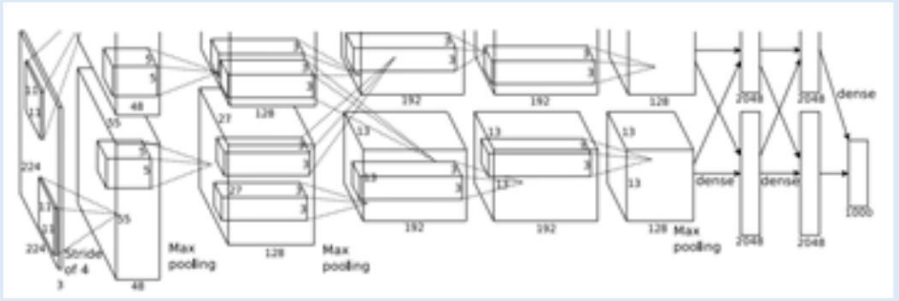


Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities





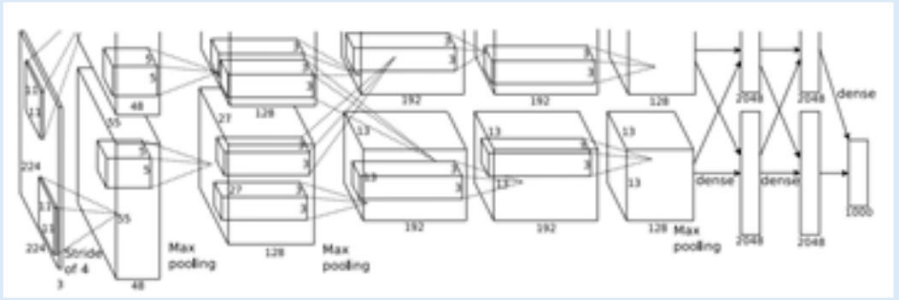
Dog

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Which pixels matter: Saliency via Backprop

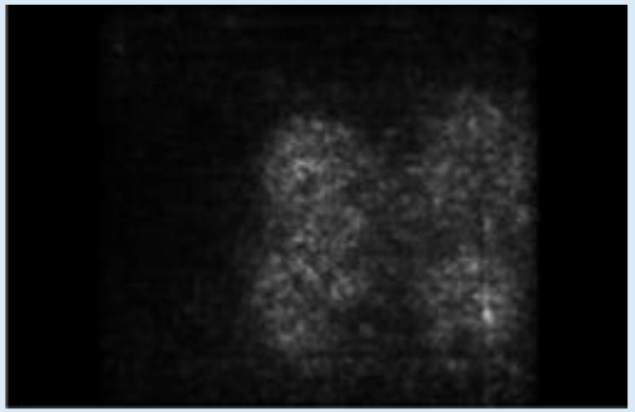
Forward pass: Compute probabilities





Dog

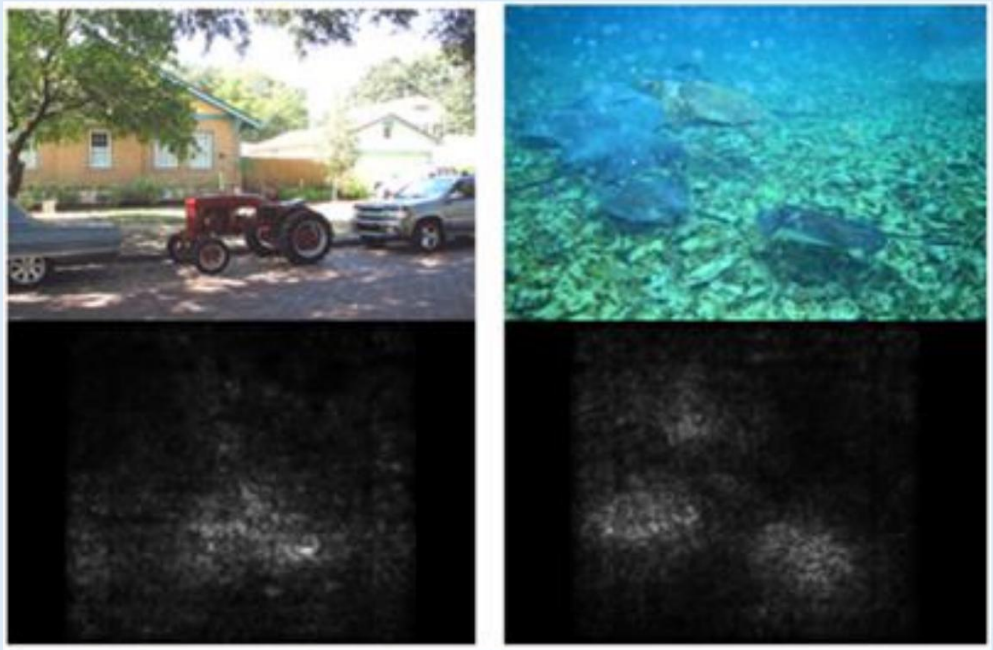
Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

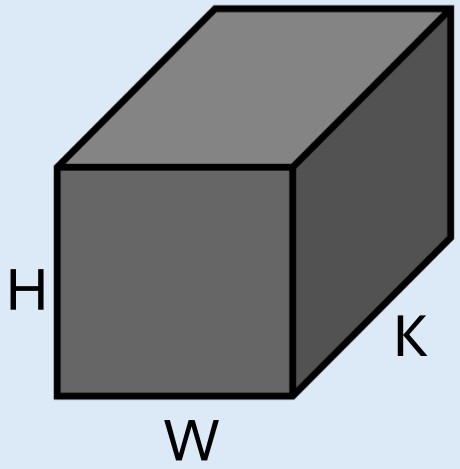
Saliency Maps



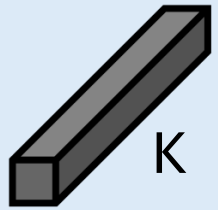


Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

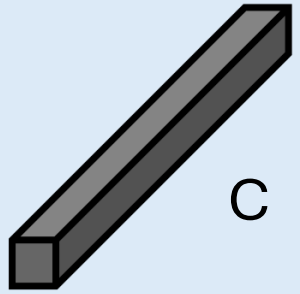
Class Activation Mapping (CAM)



Global Average Pooling



Fully Connected Layer, weights $w \in \mathbb{R}^{K \times C}$



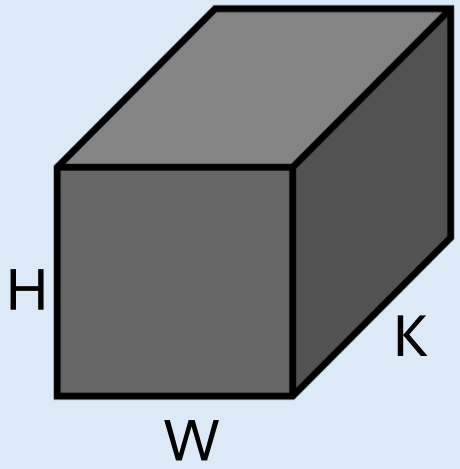
Last layer CNN features: $f \in \mathbb{R}^{H \times W \times K}$

Pooled features: $F \in \mathbb{R}^{K}$

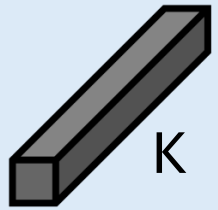
Class Scores: $S \in \mathbb{R}^{C}$

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

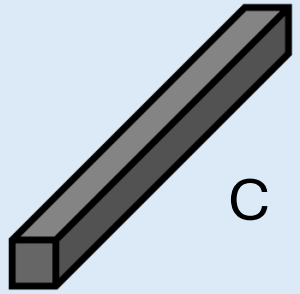
Class Activation Mapping (CAM)



Global Average Pooling



Fully Connected Layer, weights $w \in \mathbb{R}^{K \times C}$



Last layer CNN features: $f \in \mathbb{R}^{H \times W \times K}$

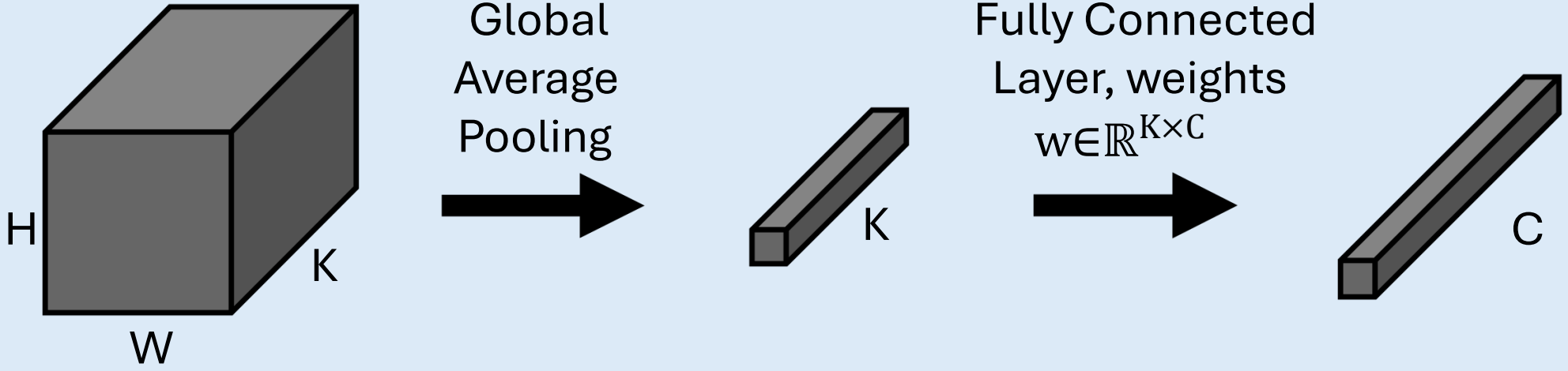
Pooled features: $F \in \mathbb{R}^{K}$

Class Scores: $S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k}$$

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

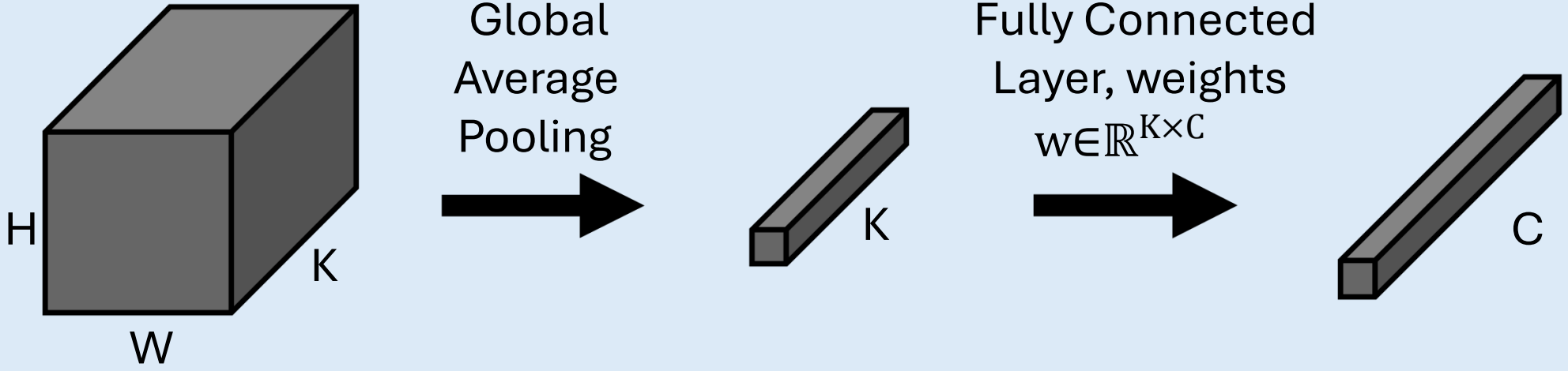
Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_k w_{k,c} F_k$$

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

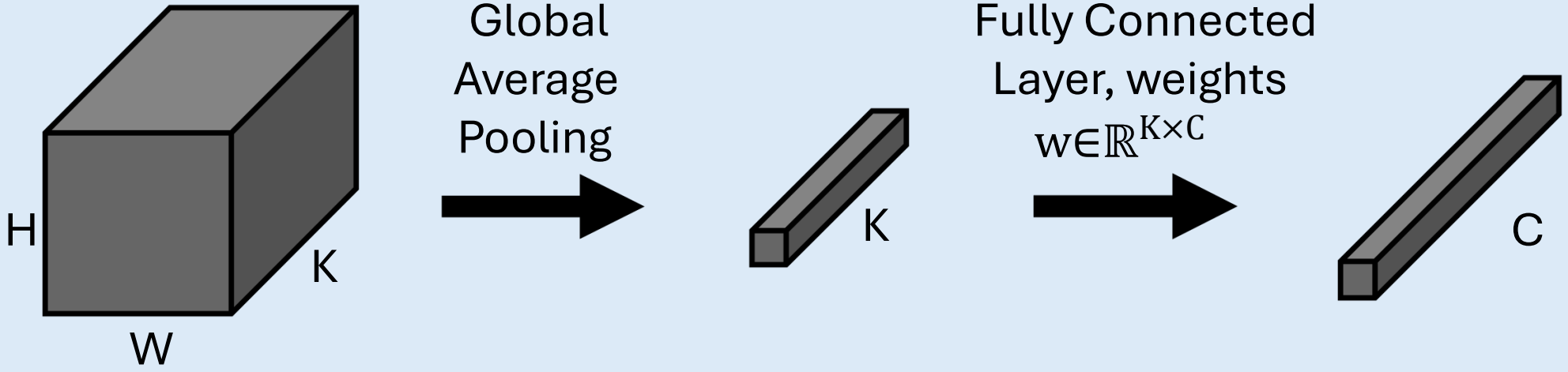
Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \qquad S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

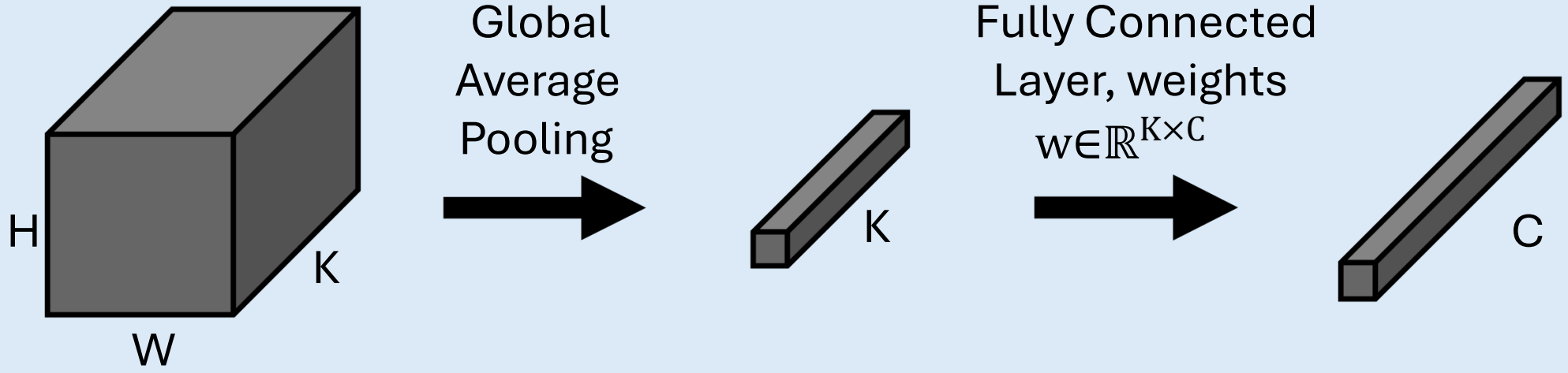
$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k}$$

$$S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

$$= \frac{1}{HW} \sum_{h,w} \sum_k w_{k,c} f_{h,w,k}$$

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

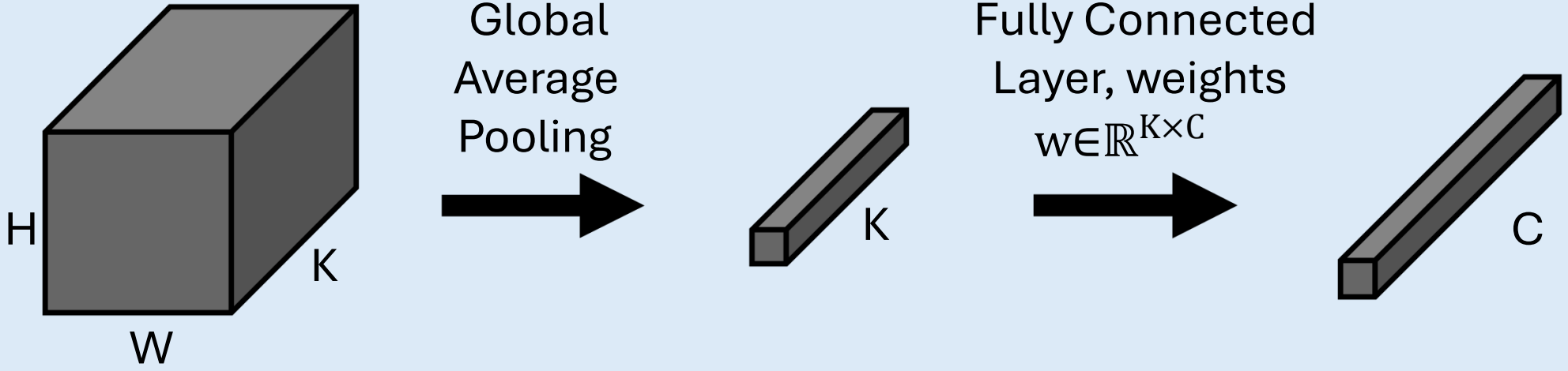
$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k}$$

$$S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

$$= \frac{1}{HW} \sum_{h,w} \sum_k w_{k,c} f_{h,w,k}$$

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class Activation Mapping (CAM)



Last layer CNN features:
$f \in \mathbb{R}^{H \times W \times K}$

Pooled features:
$F \in \mathbb{R}^{K}$

Class Scores:
$S \in \mathbb{R}^{C}$

$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k}$$

$$S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

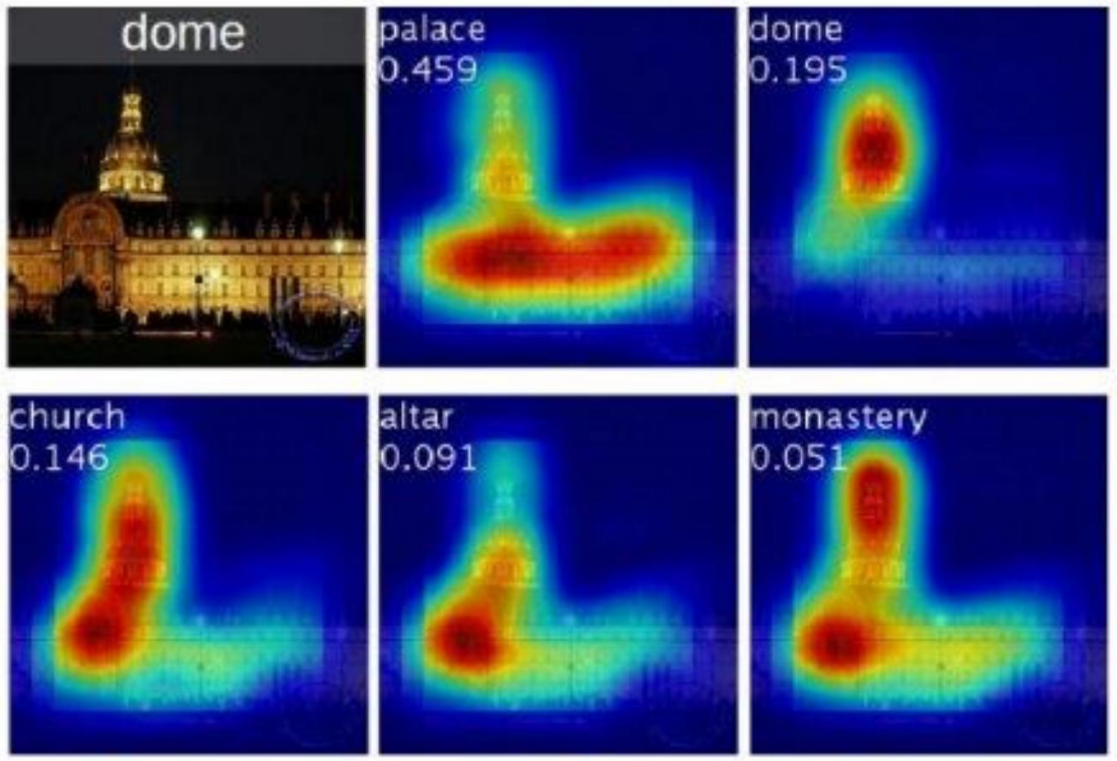
$$= \frac{1}{HW} \sum_{h,w} \sum_k w_{k,c} f_{h,w,k}$$

Class Activation Maps:
$M \in \mathbb{R}^{C \times H \times W}$

$$M_{c,h,w} = \sum_k w_{k,c} f_{h,w,k}$$

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class Activation Mapping (CAM)



Class activation maps of top 5 predictions

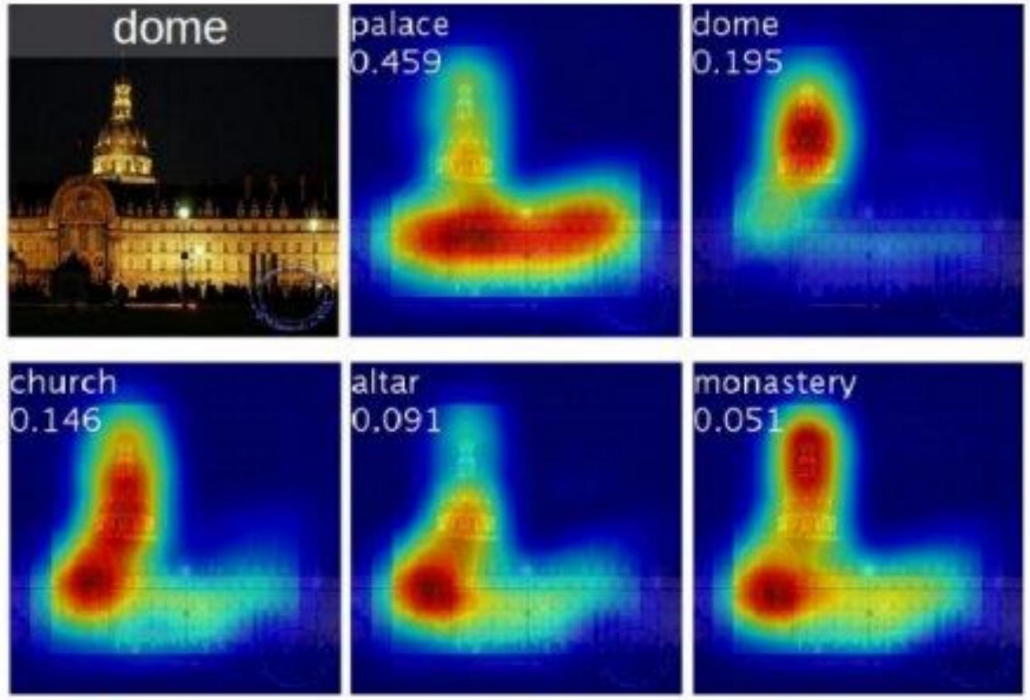


Class activation maps for one object class

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class Activation Mapping (CAM)

Problem: Can only apply to last conv layer



Class activation maps of top 5 predictions



Class activation maps for one object class

Zhou et al, “Learning Deep Features for Discriminative Localization”, CVPR 2016

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$

Selvaraju et al, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”, CVPR 2017



Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A:

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

Selvaraju et al, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”, CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A:

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

3. Global Average Pool the gradients to get weights $\alpha \in \mathbb{R}^{K}$:

$$\alpha_k = \frac{1}{HW} \sum_{h,w} \frac{\partial S_c}{\partial A_{h,w,k}}$$

Selvaraju et al, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”, CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A:

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

3. Global Average Pool the gradients to get weights $\alpha \in \mathbb{R}^{K}$:

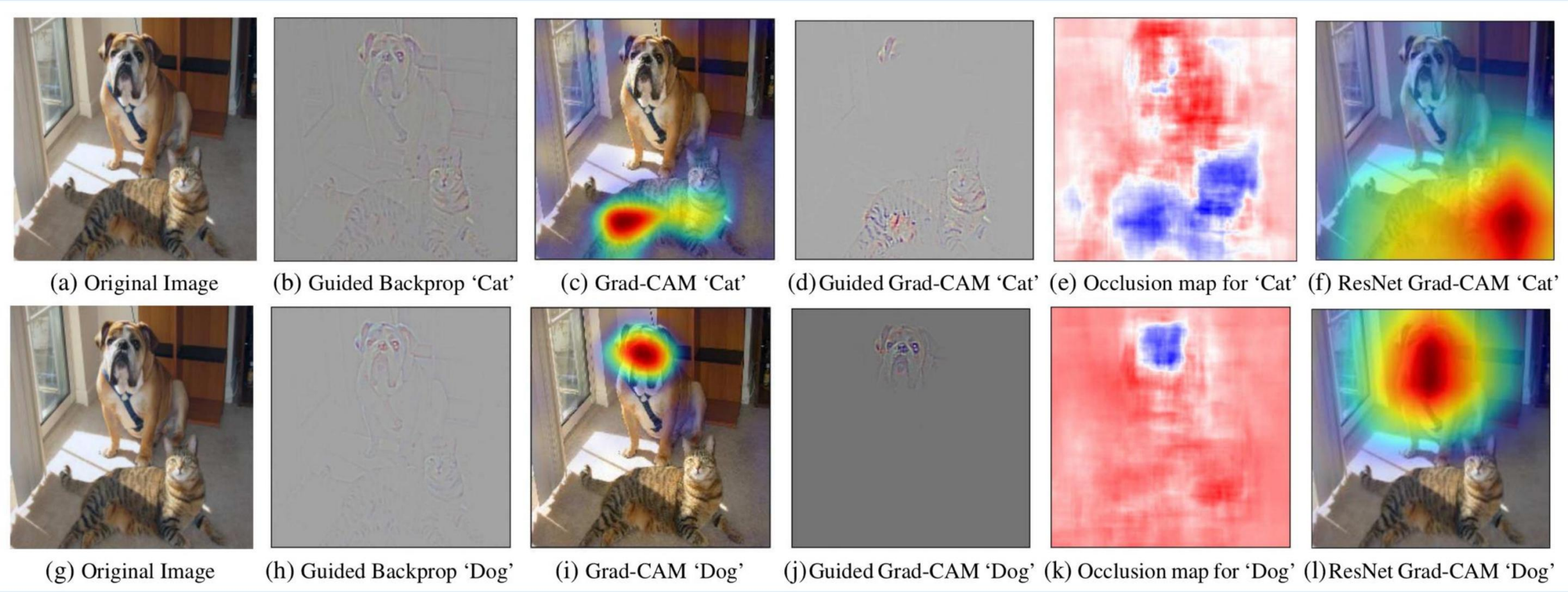
$$\alpha_k = \frac{1}{HW} \sum_{h,w} \frac{\partial S_c}{\partial A_{h,w,k}}$$

4. Compute activation map $M^c \in \mathbb{R}^{H,W}$:

$$M_{h,w}{}^{c} = \mathrm{ReLU}\left(\sum_k \alpha_k A_{h,w,k} \right)$$

Selvaraju et al, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”, CVPR 2017

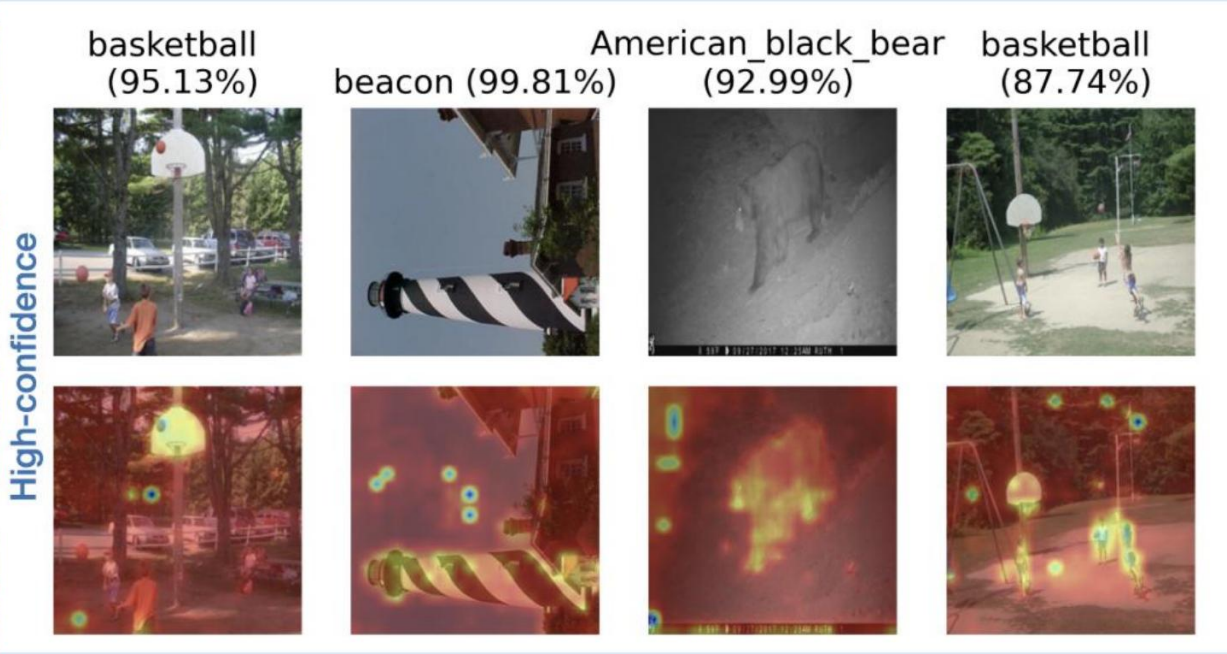
Gradient-Weighted Class Activation Mapping (Grad-CAM)



Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Visualizing ViT features





Chen et al., When Vision Transformers Outperform Resnets Without Pre-training Or Strong Data Augmentations, ICLR 2022; Paul and Chen, Vision Transformers are Robust Learners, AAAI 2022. Reproduced for educational purposes.

Today

- Transformers Recap

- **Computer Vision Tasks**
  - Semantic Segmentation
  - Object Detection
  - Instance Segmentation

- Visualization & Understanding
  - Model Layers Visualization
  - Saliency Maps
  - CAM & Grad-CAM

References

- Stanford CS231n

- Mathematical Foundations of Deep Neural Networks at Seoul National University.

