Machine Learning

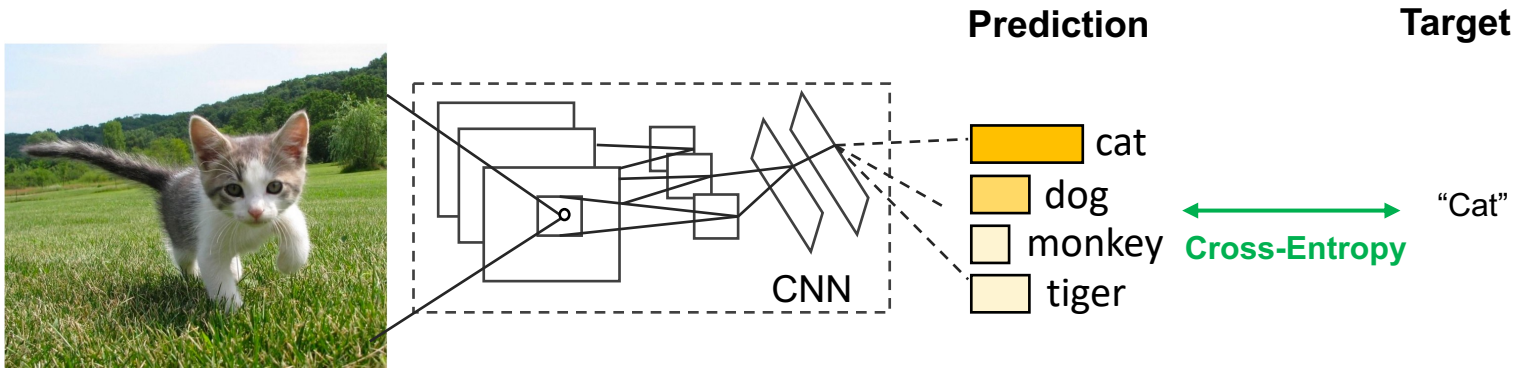
Lecture 14: Computer Vision

Fall 2023

Instructor: Xiaodong Gu

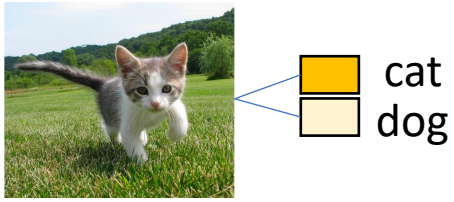
Recall: Image Classification

- Convolution + MLP + Softmax

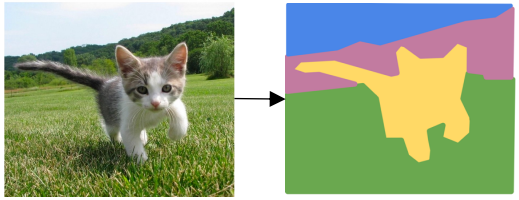


Computer Vision Tasks

Classification



Semantic Segmentation



Object Detection

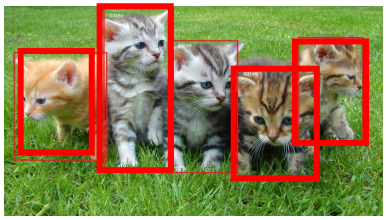


Image Captioning

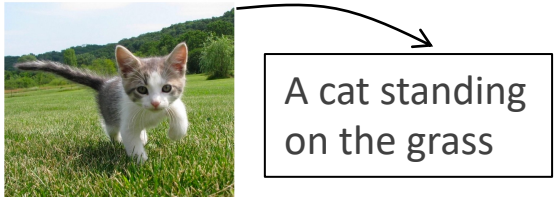
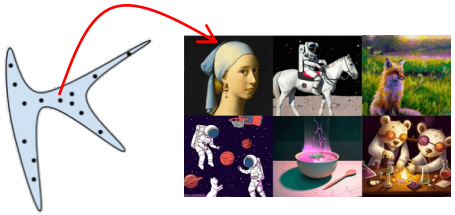


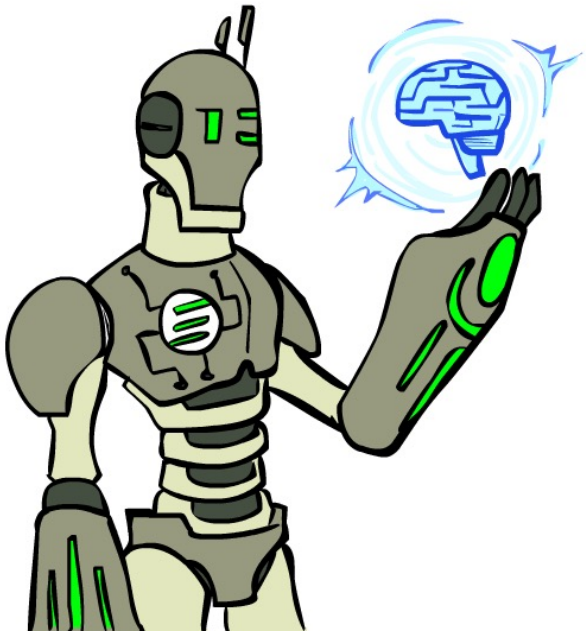
Image Generation



.....

Today

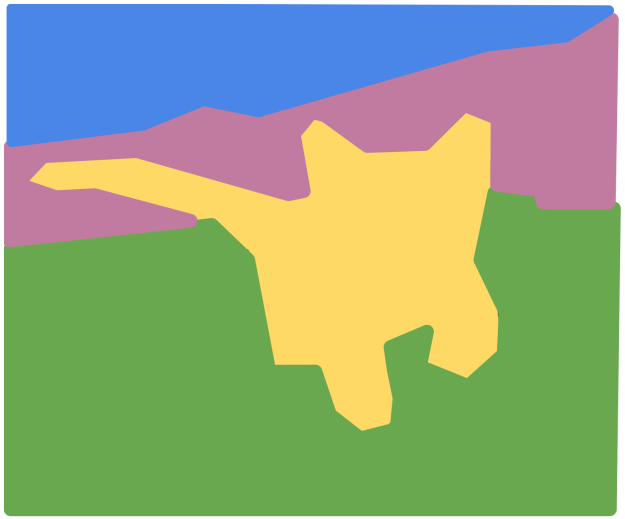
- Image Segmentation
- Object Detection
- Image Captioning



Semantic Segmentation

- Segment an image into disjoint regions that have independent semantics.

Label each pixel in the image with a category



GRASS,
CAT,
TREE,
SKY, ...

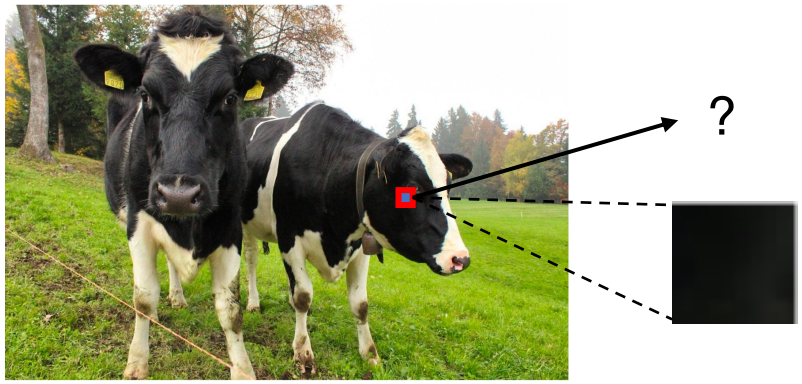
Input: an image

Output: class of each pixel

A Naïve Idea

- **Classifying Pixels?**

Full image

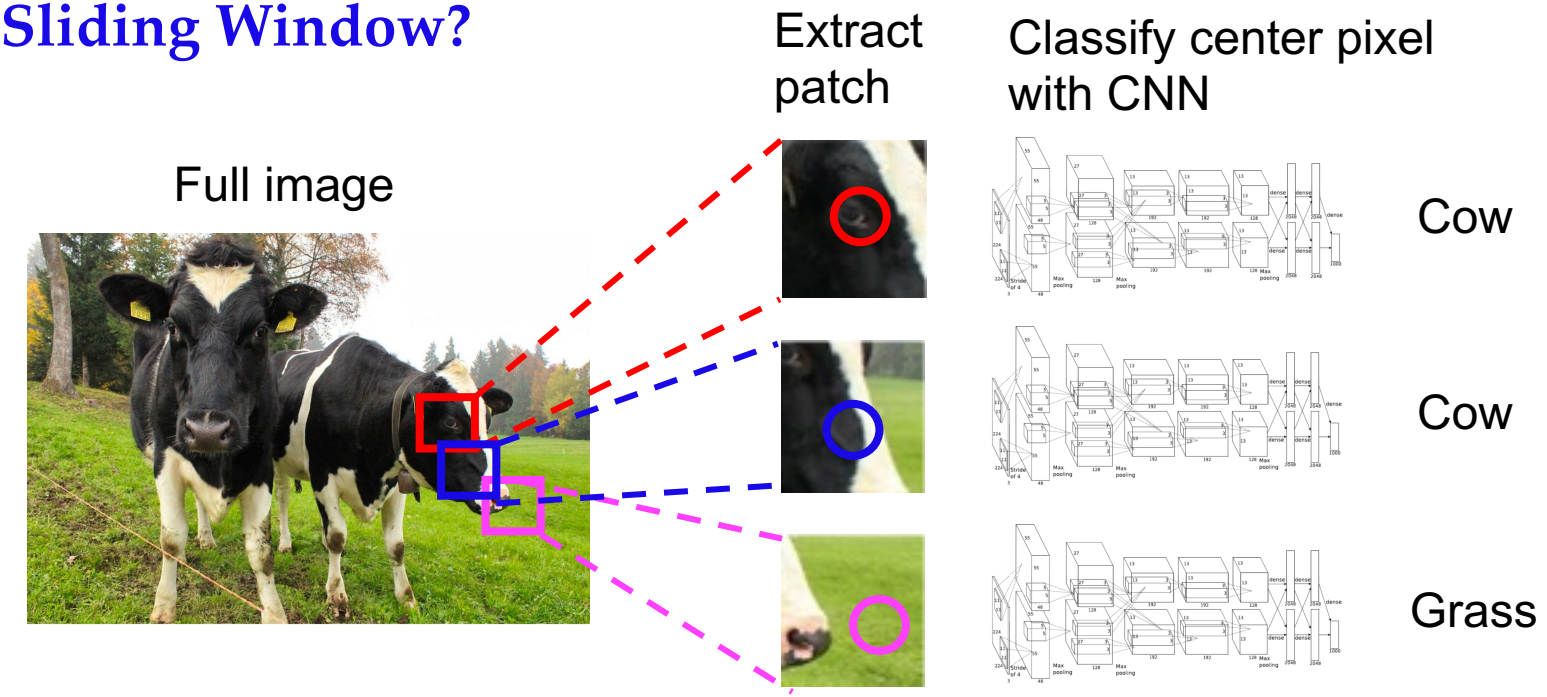


Impossible to classify without context

Q: how do we include context?

The Idea

- **Sliding Window?**

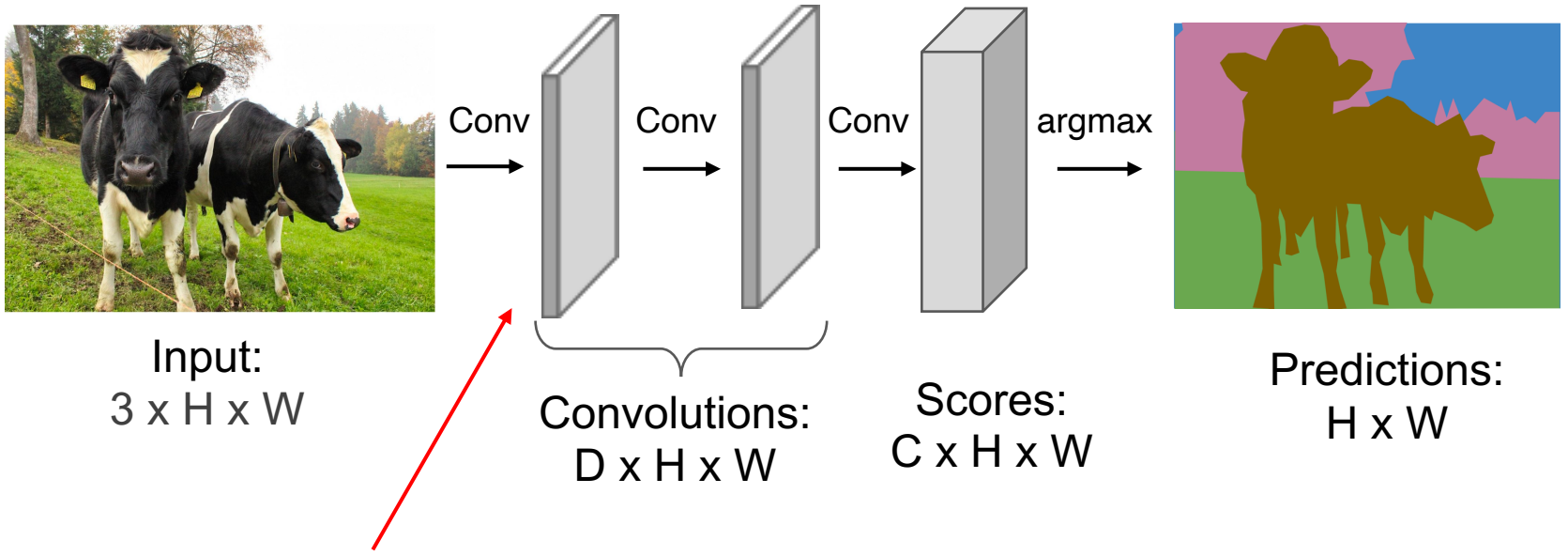


Problem: Very inefficient! Not reusing shared features between overlapping patches

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

Fully Convolutional?

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

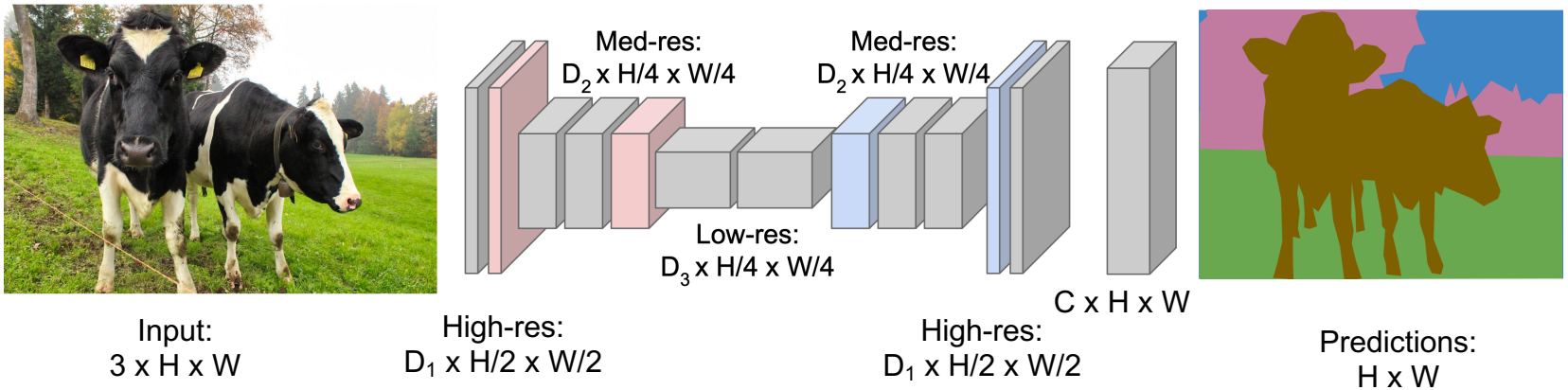


Fully Convolutional?

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

Downsampling:
Pooling, strided convolution

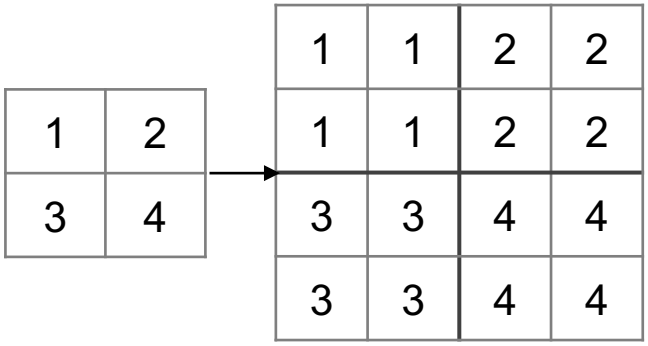
Upsampling:
???



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

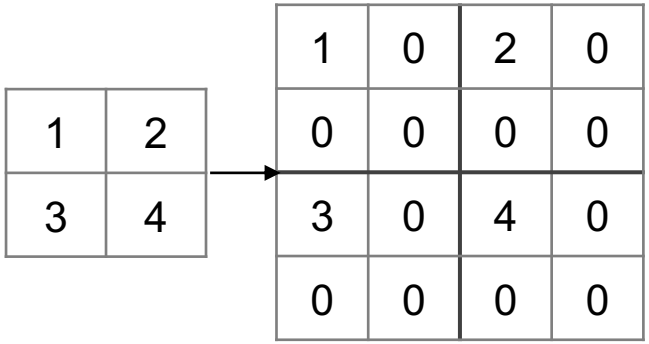
In-Network Upsampling: "Unpooling"

Nearest Neighbor



Input: 2 x 2 Output: 4 x 4

"Bed of Nails"



Input: 2 x 2 Output: 4 x 4

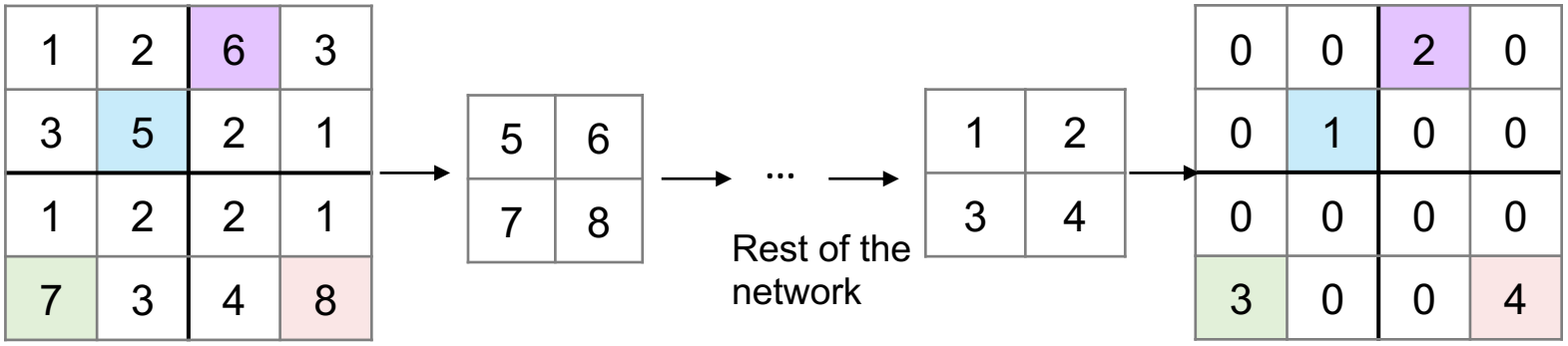
In-Network Upsampling: "Max Unpooling"

Max Pooling

Remember which element was max!

Max Unpooling

Use positions from pooling layer



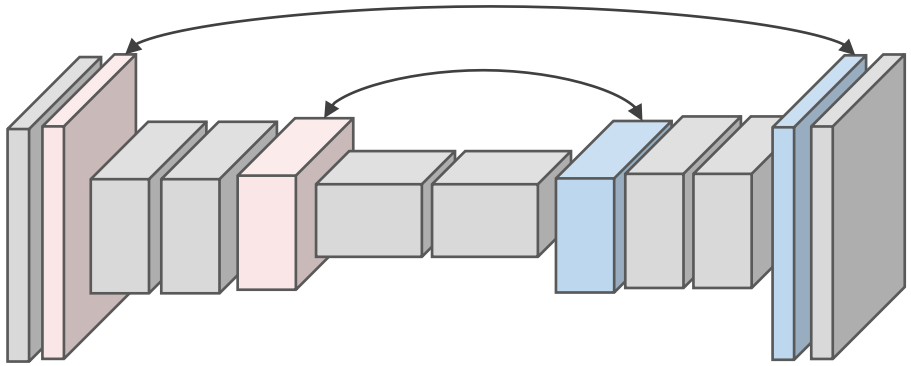
Input: 4 x 4

Output: 2 x 2

Input: 2 x 2

Output: 4 x 4

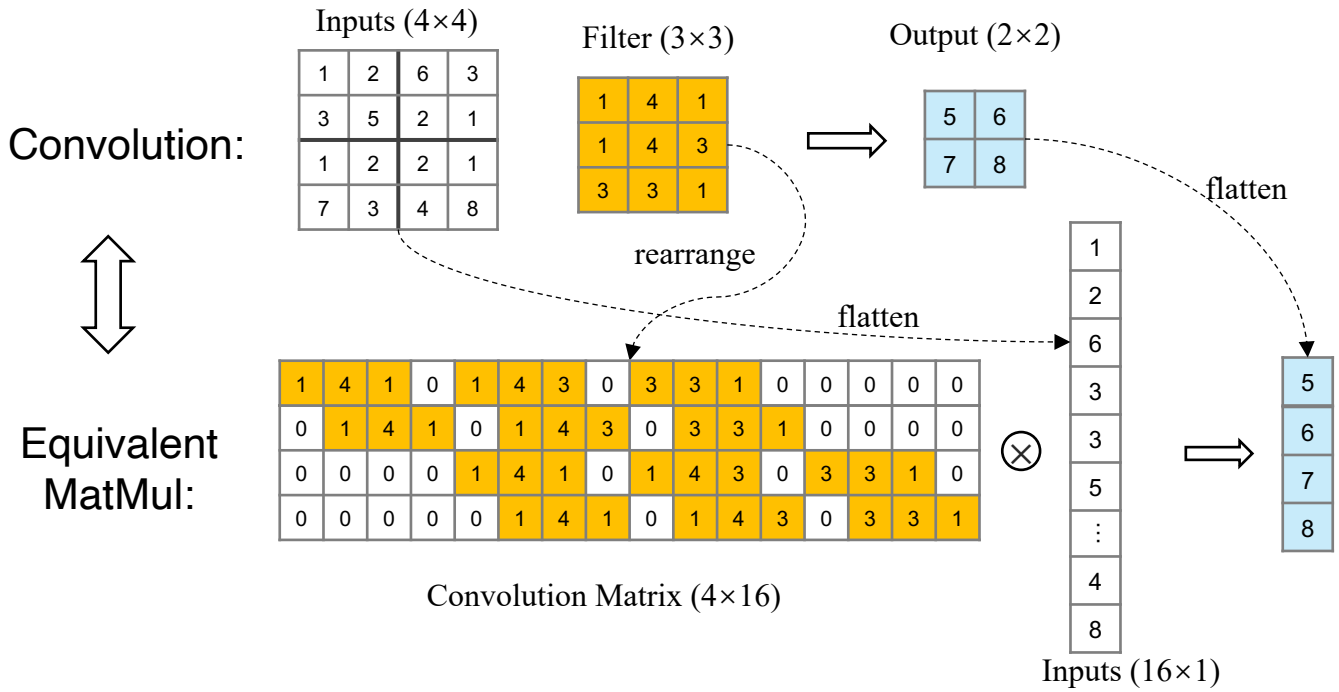
Corresponding pairs of downsampling and upsampling layers



Upsampling with Transposed Convolution

- **Convolution Matrix**

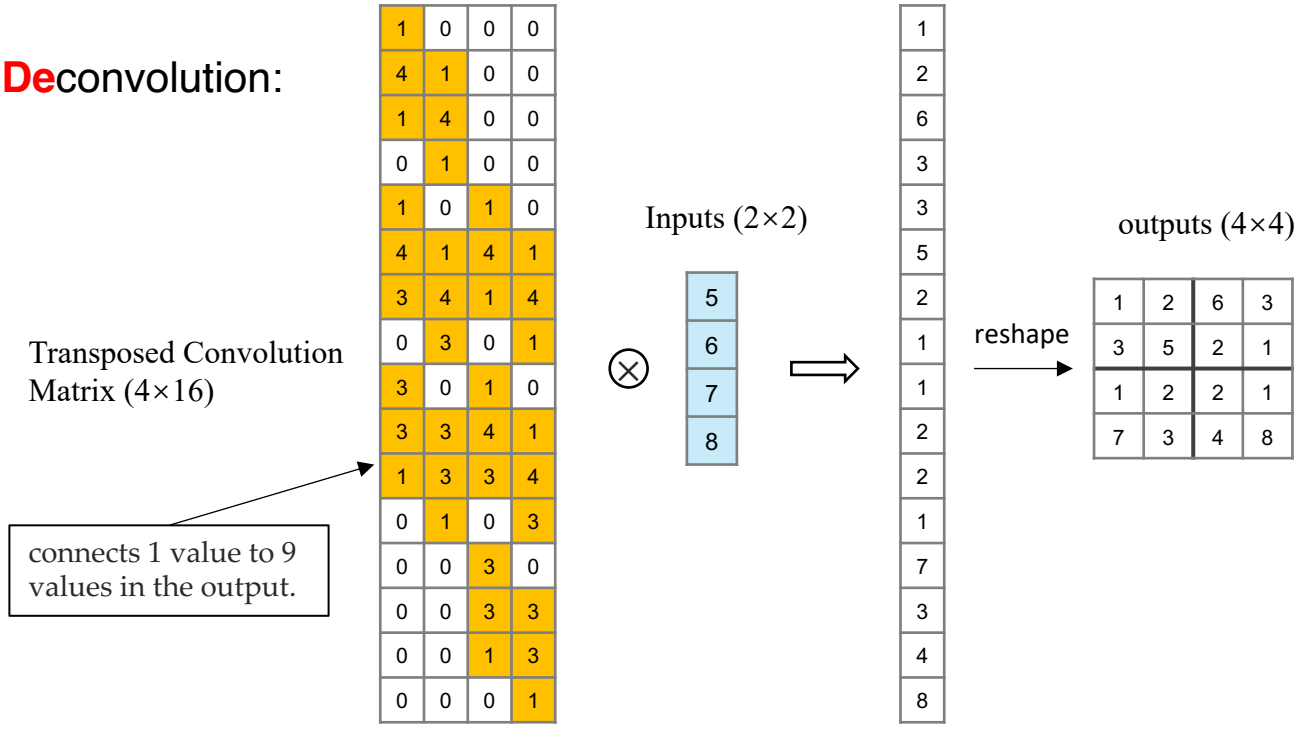
  We can express a convolution operation using a **matrix multiplication** by rearranging the kernel matrix into a general matrix.



Upsampling with Transposed Convolution

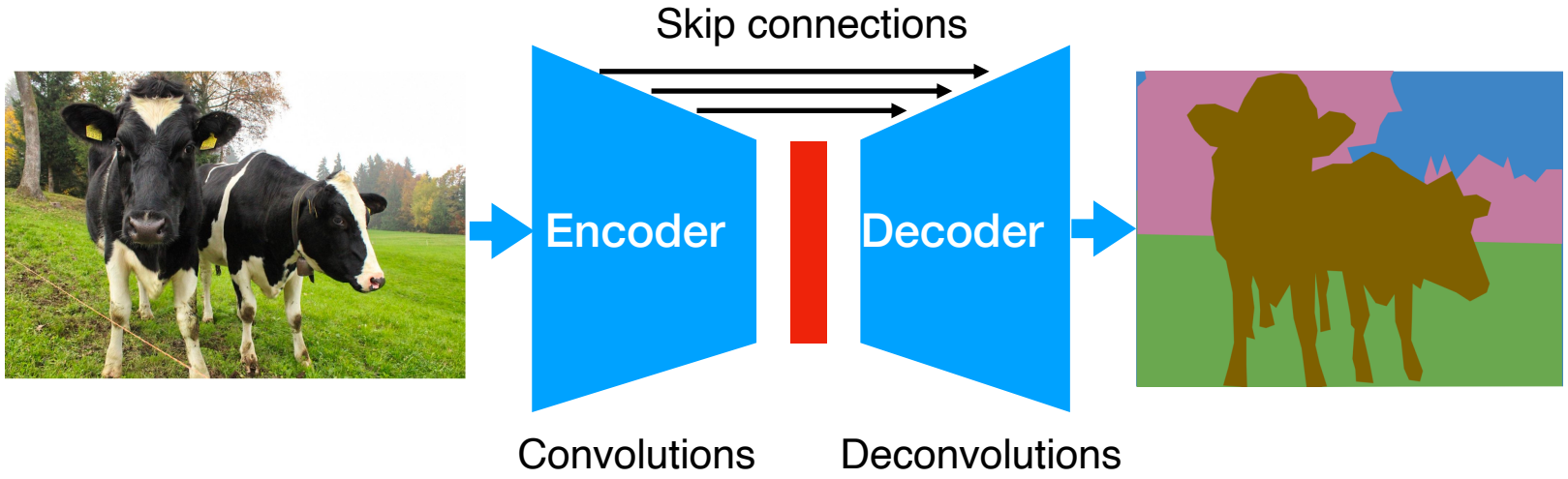
- **Transposed Convolution Matrix**

  Transpose the convolution matrix $\boldsymbol{C}$ (4×16) to $\boldsymbol{C}^{\mathrm{T}}$ (16×4).
  We can matrix-multiply $\boldsymbol{C}^{\mathrm{T}}$ (16×4) with a column vector (4×1) to generate an output matrix (16×1).



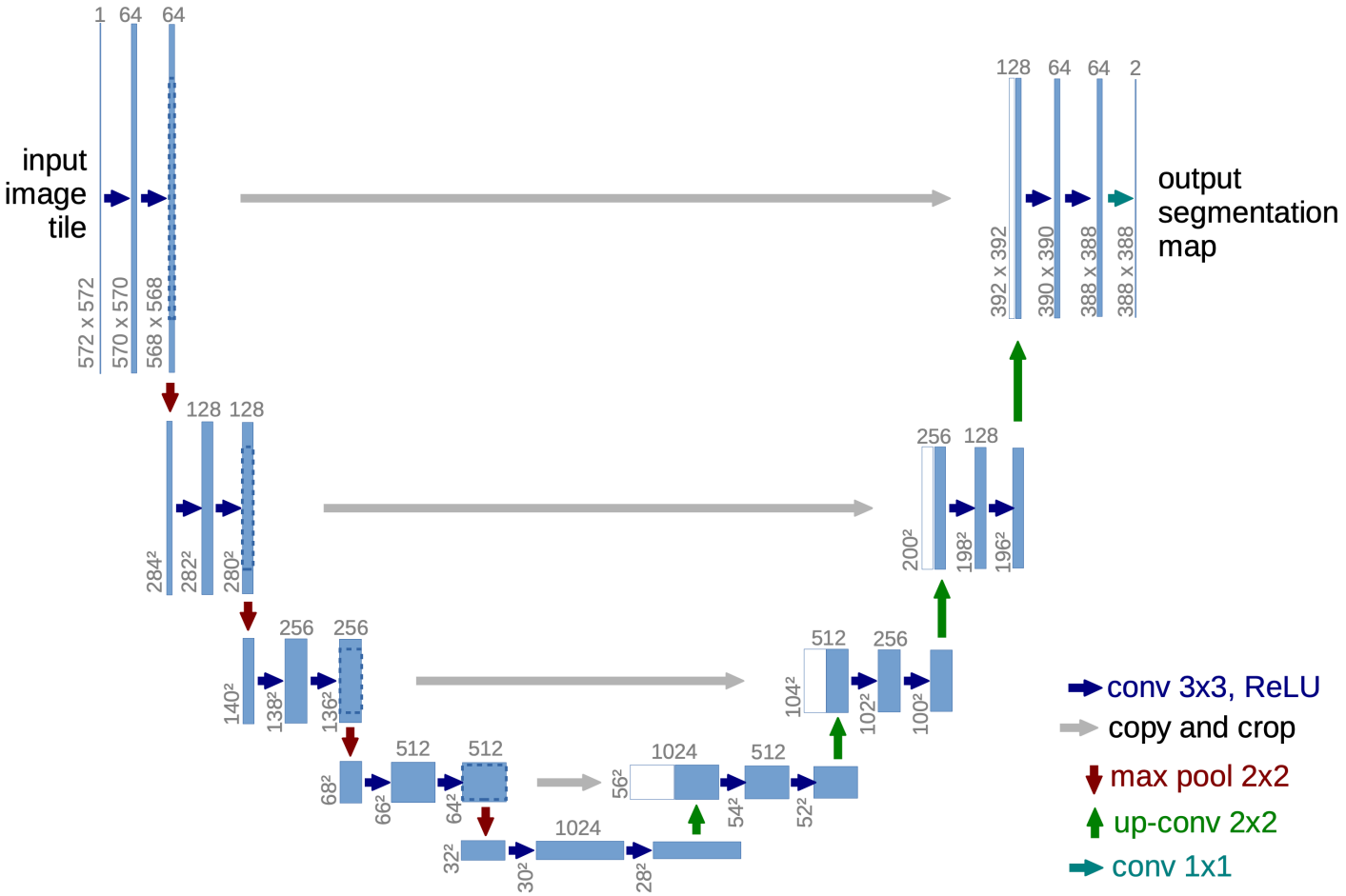
Encoder-Decoder Architectures

- U-Net



Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation

U-Net

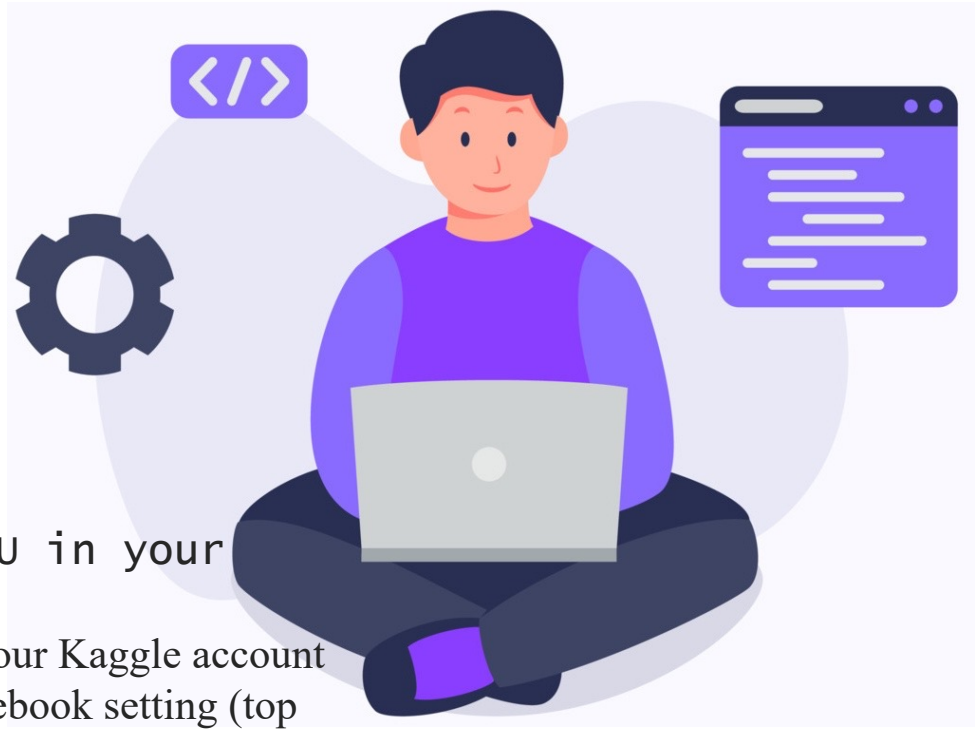


Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation

TIME for Coding

- **Tutorial**: **Image segmentation with U-Net**

  https://www.kaggle.com/code/gokulkarthik/image-segmentation-with-unet-pytorch

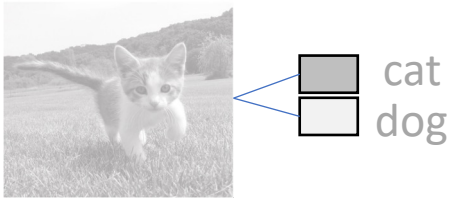


NOTE: How to enable GPU in your Kaggle kernel?

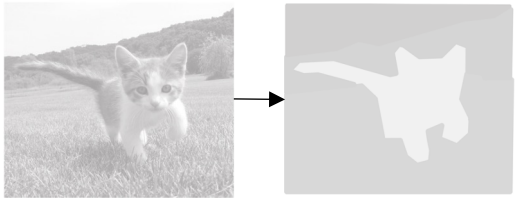
- Verify the phone number of your Kaggle account
- Click "accelerator" in the notebook setting (top right) and then select the GPU (https://www.kaggle.com/general/97939)

Computer Vision Tasks

Classification



Semantic Segmentation



Object Detection

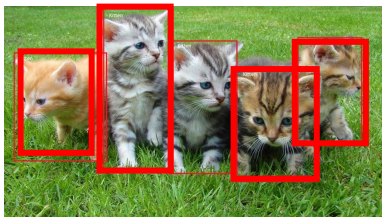
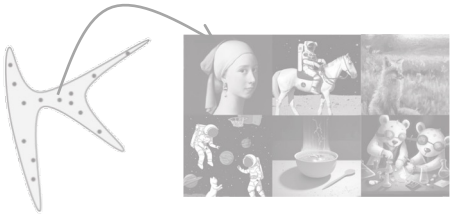
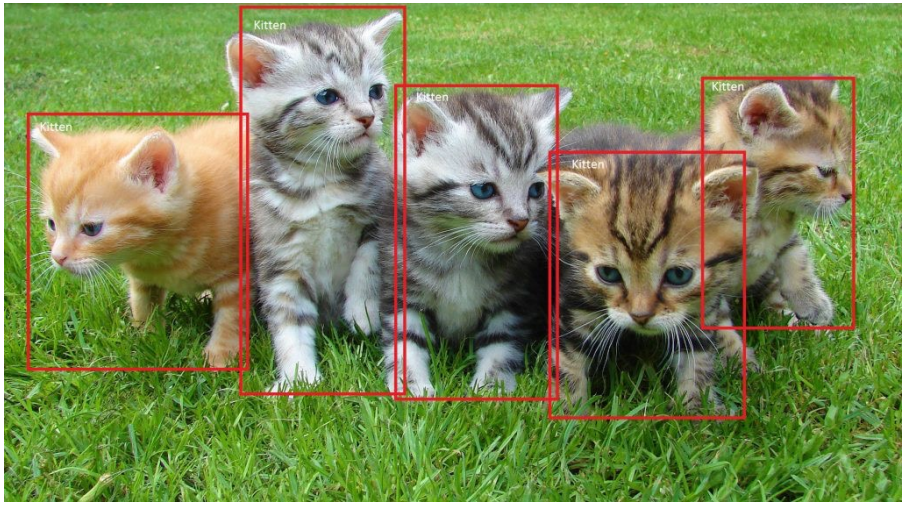


Image Captioning

Image Generation

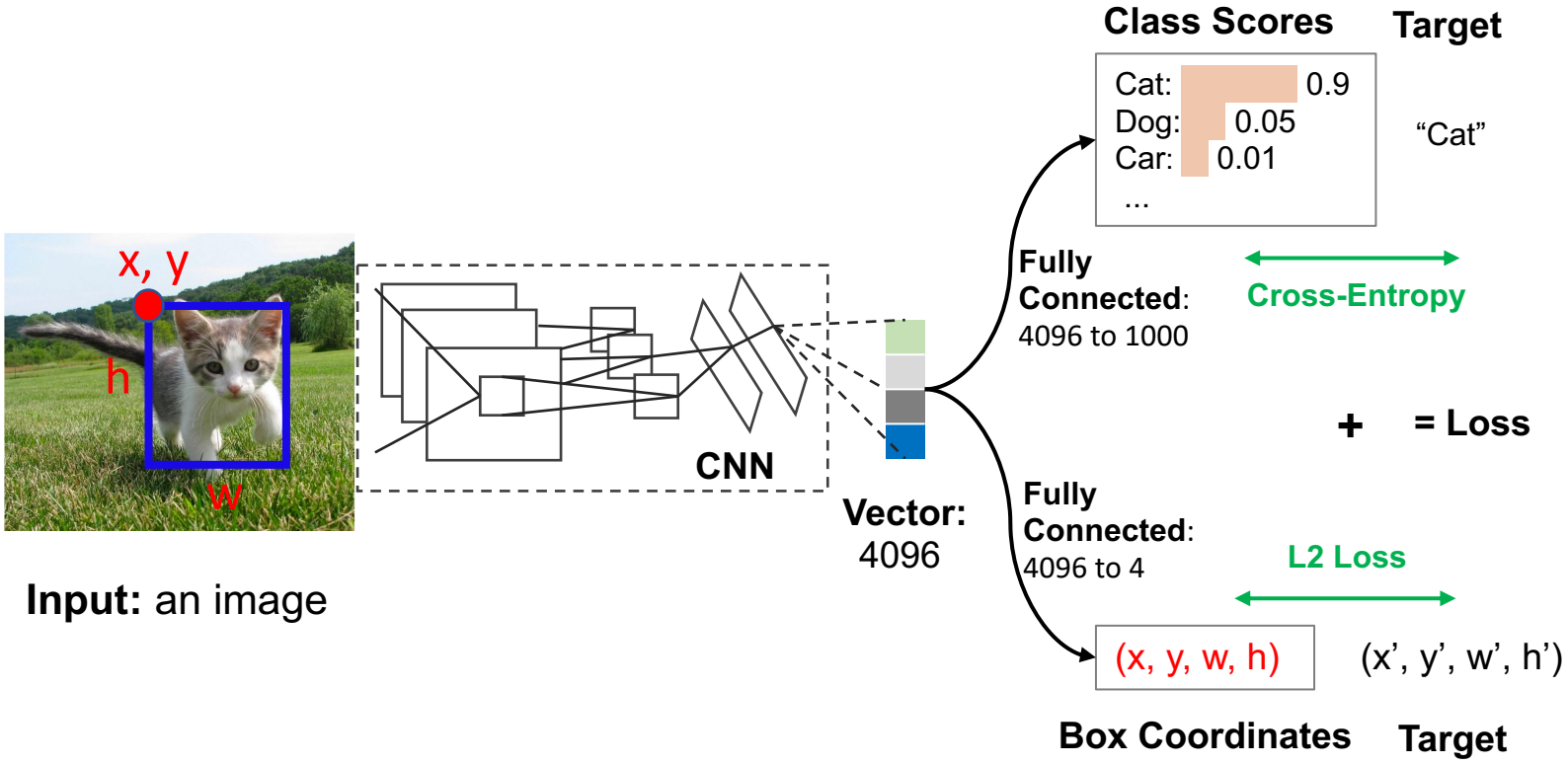


Object Detection

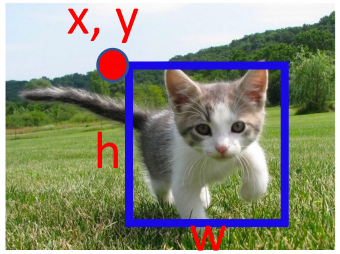


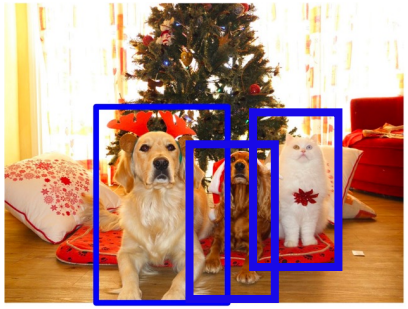
Single Object

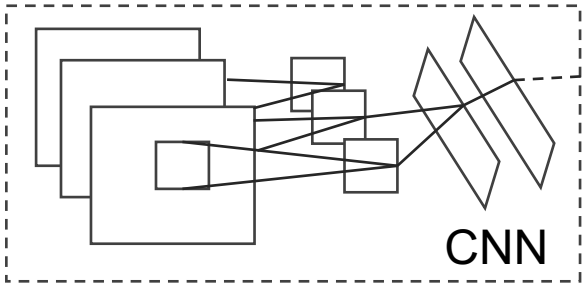
- **Object detection = Classification + Localization**



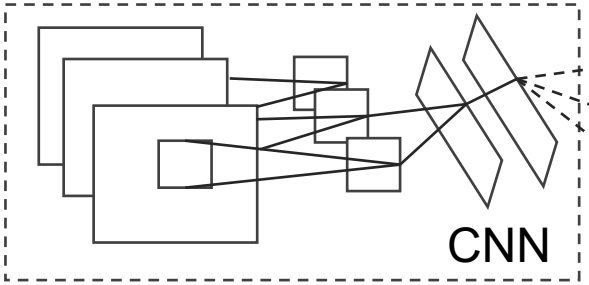
Multiple Objects?



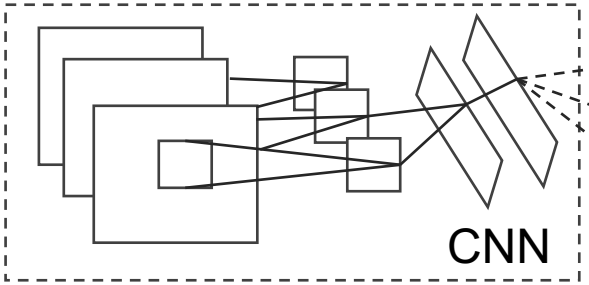




CAT: (x, y, w, h)



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



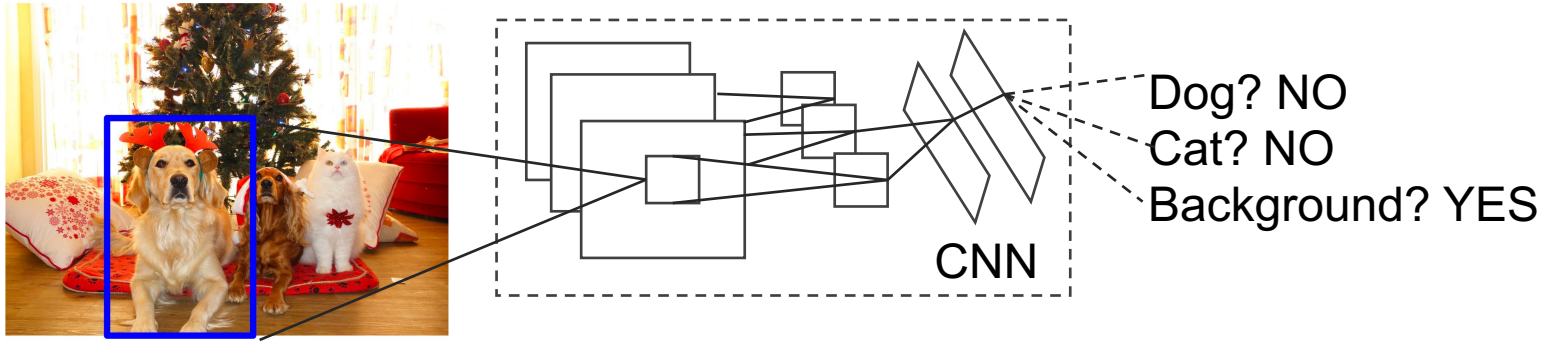
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

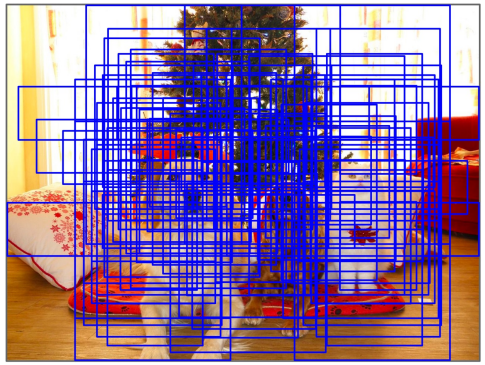
Too many predictions

- **Problem:** each image (CNN) needs a different number of outputs!

Multiple Objects

- Idea: enumerate different crops of the image, apply a CNN to classify **each** crop as object or background.

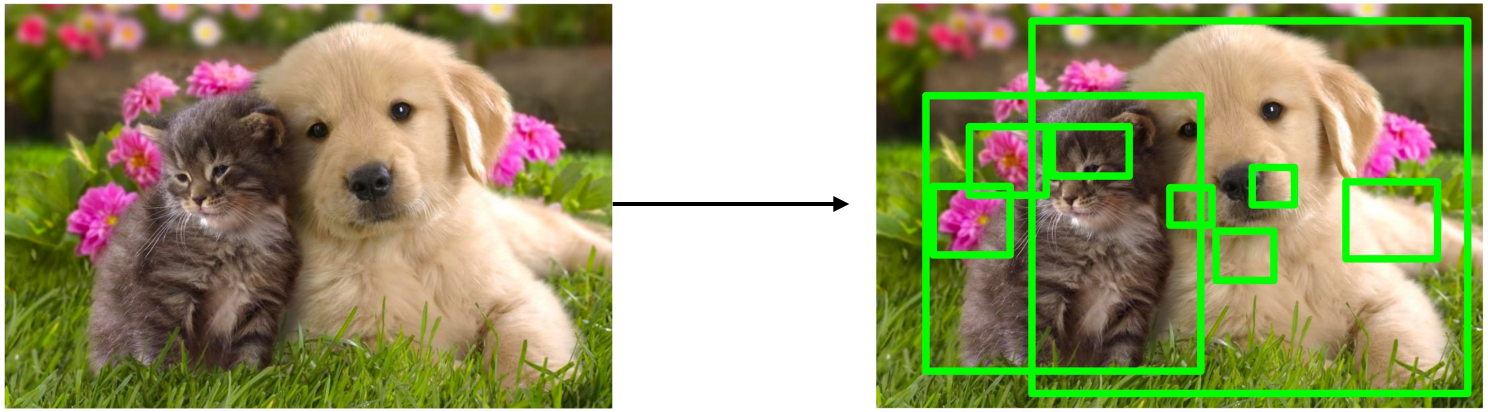




Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Region Proposals: Selective Search

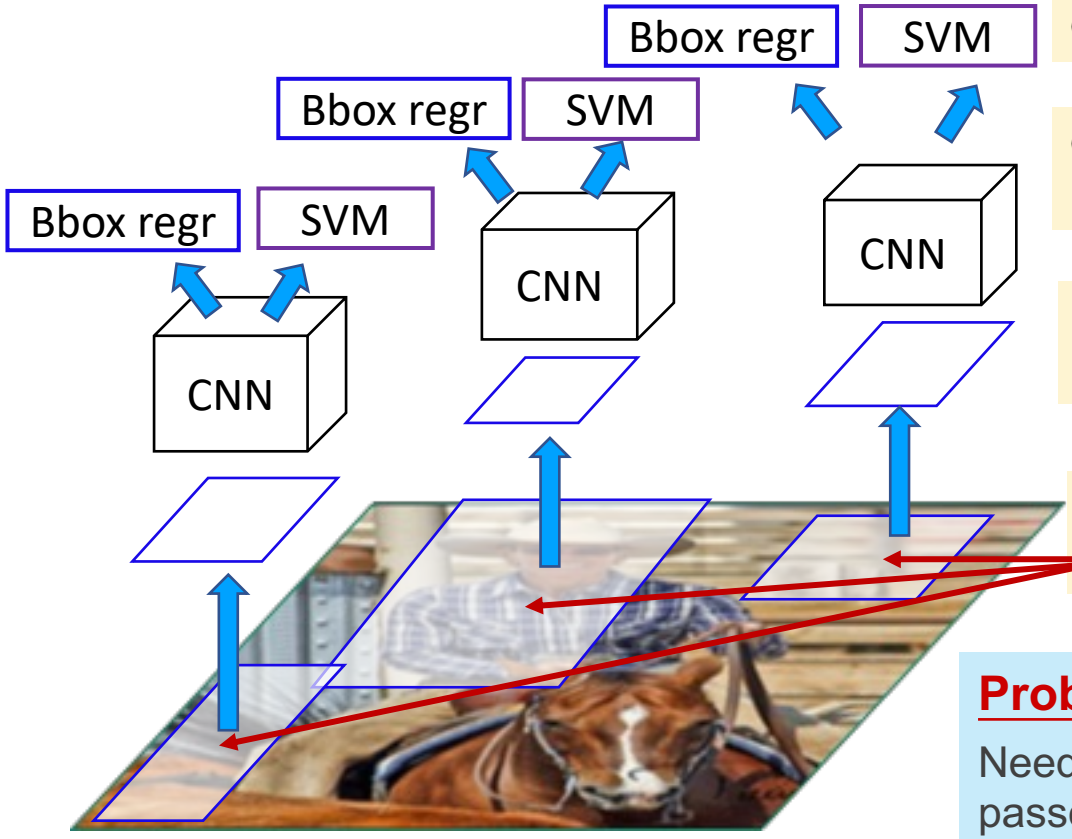
- Find "blobby" image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



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

⑤ Predict “corrections” to the ROI position: (dx, dy, dw, dh)



④ Classify regions with SVMs

③ Forward each region through ConvNet (ImageNet-pretrained)

② Warped image regions (224x224 pixels)

① **Regions of Interest** (ROI) from a proposal method (~2k)

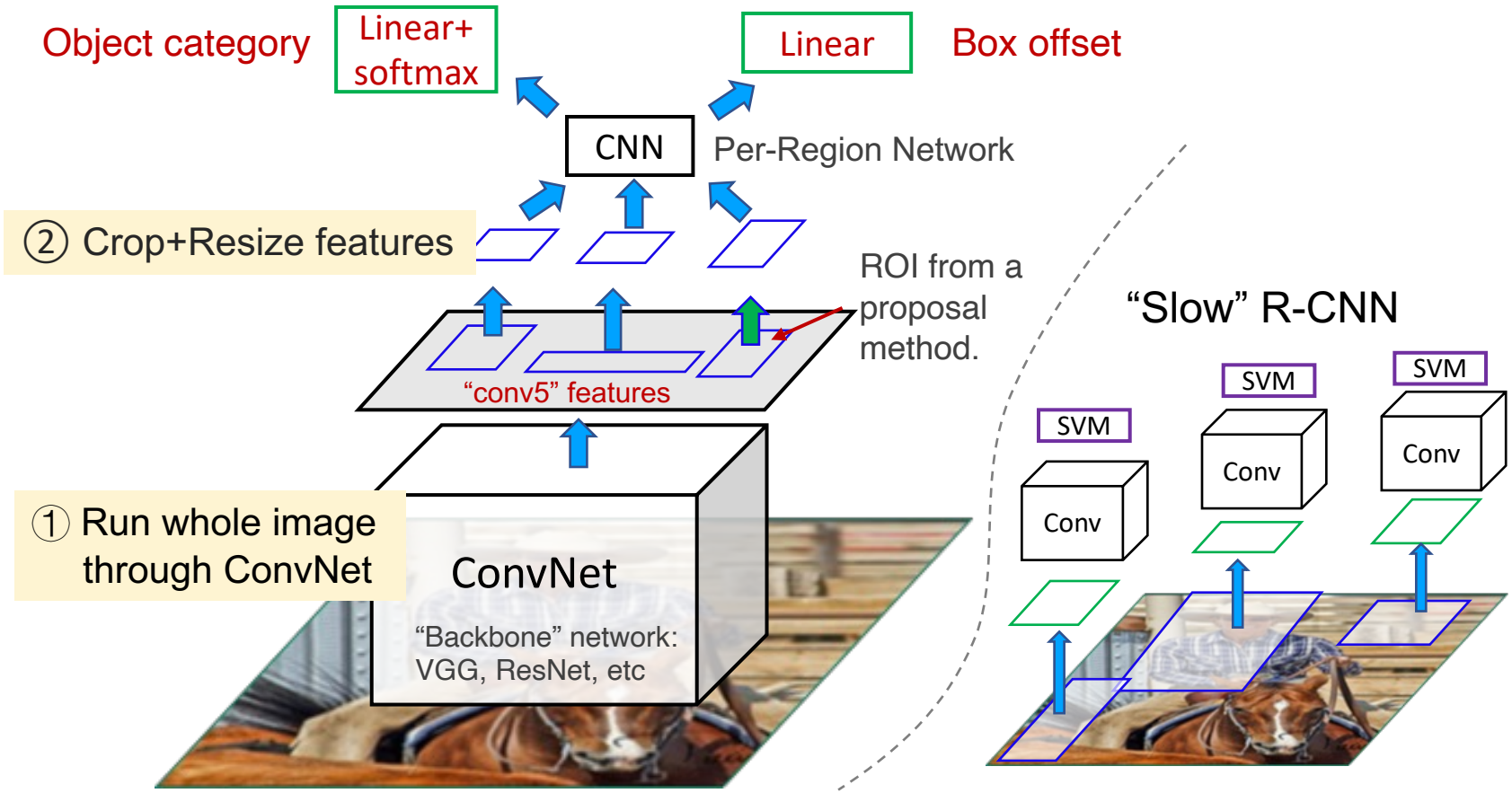
Problem: Very slow!

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.

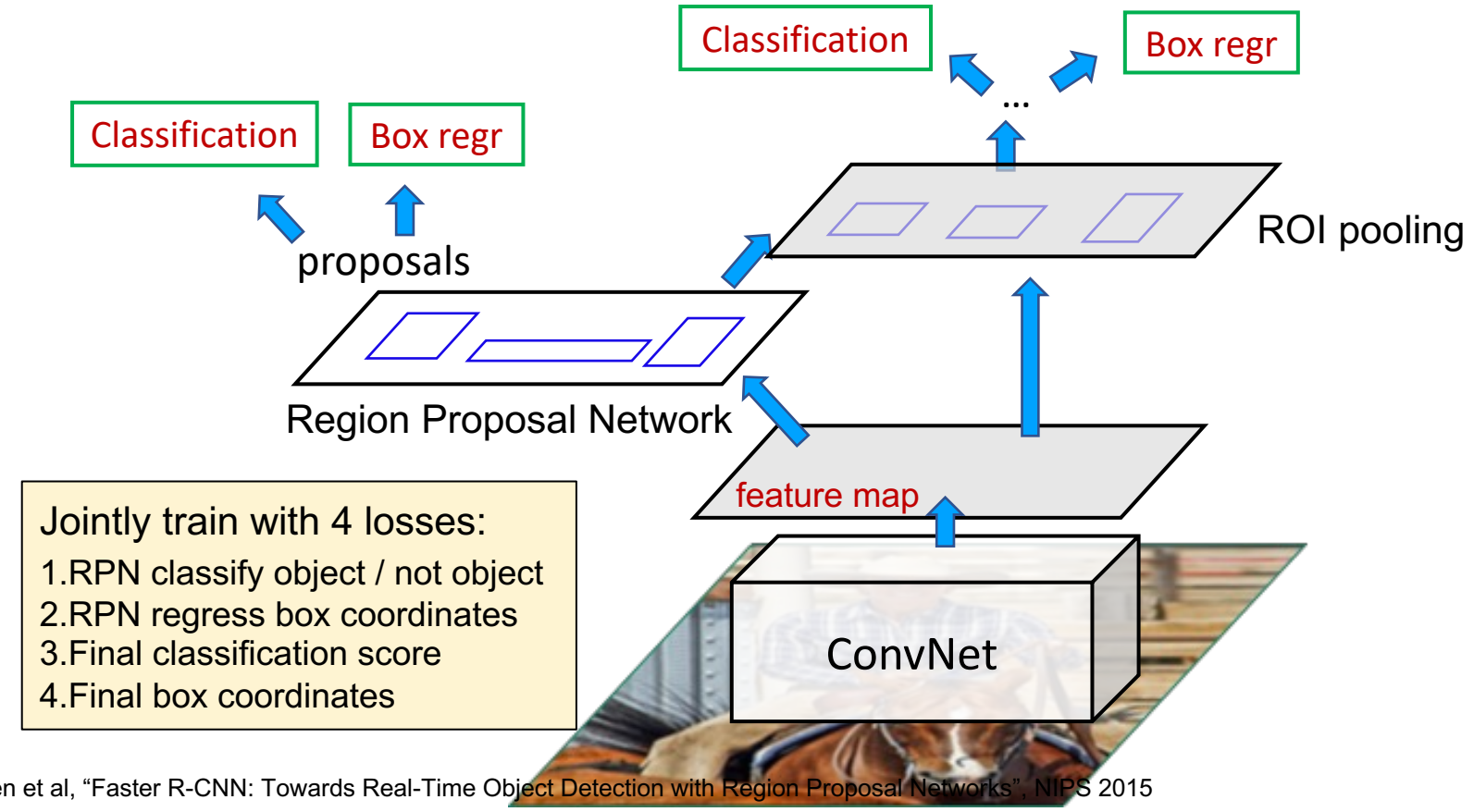
Fast R-CNN Crop the conv feature instead!

- **Idea:** Pass the image through convnet before cropping!



Faster R-CNN Make CNN do proposals !

- Insert a Region Proposal Network (RPN) to predict proposals from features



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

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)

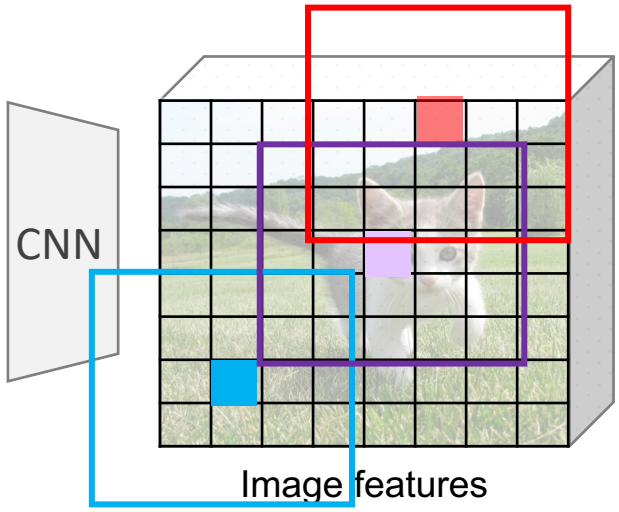
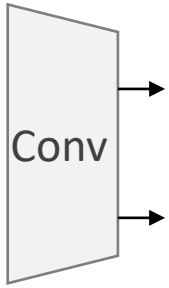


Image features
(e.g. 512 x 20 x 15)

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



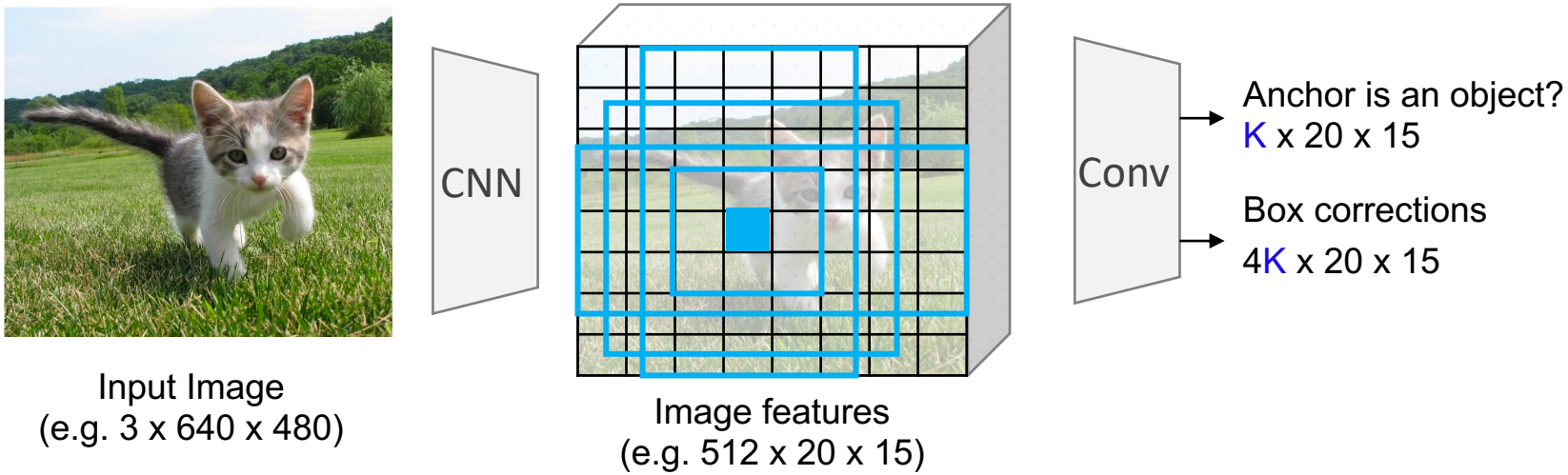
Anchor is an object?
1 x 20 x 15

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

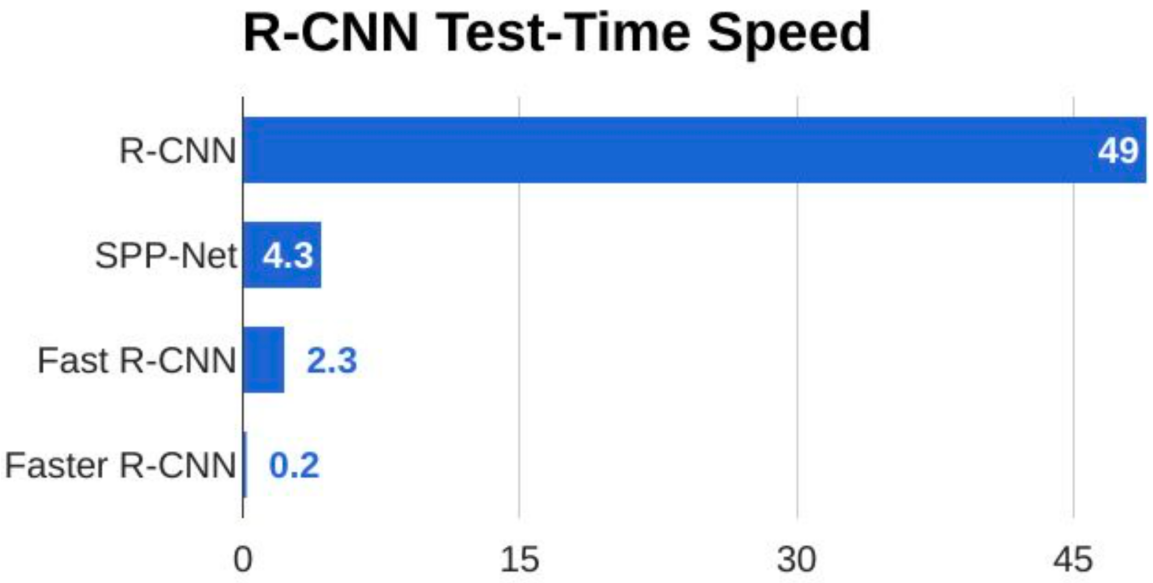
- In practice use K different anchor boxes of different size / scale at each point



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

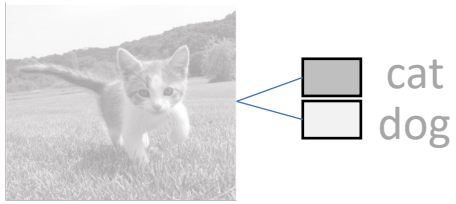
Faster R-CNN

- How fast?

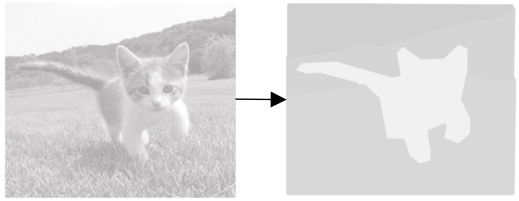


Computer Vision Tasks

Classification



Semantic Segmentation



GRASS, CAT,
TREE, SKY, ...

Object Detection

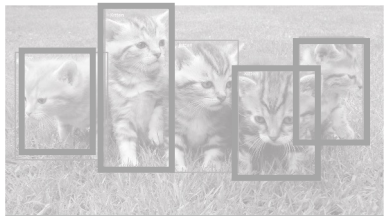


Image Captioning

Image Generation

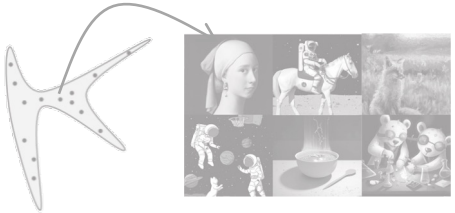


Image Captioning

- Generate a natural language description for an image.

"A person wearing a hat is sitting by the river"

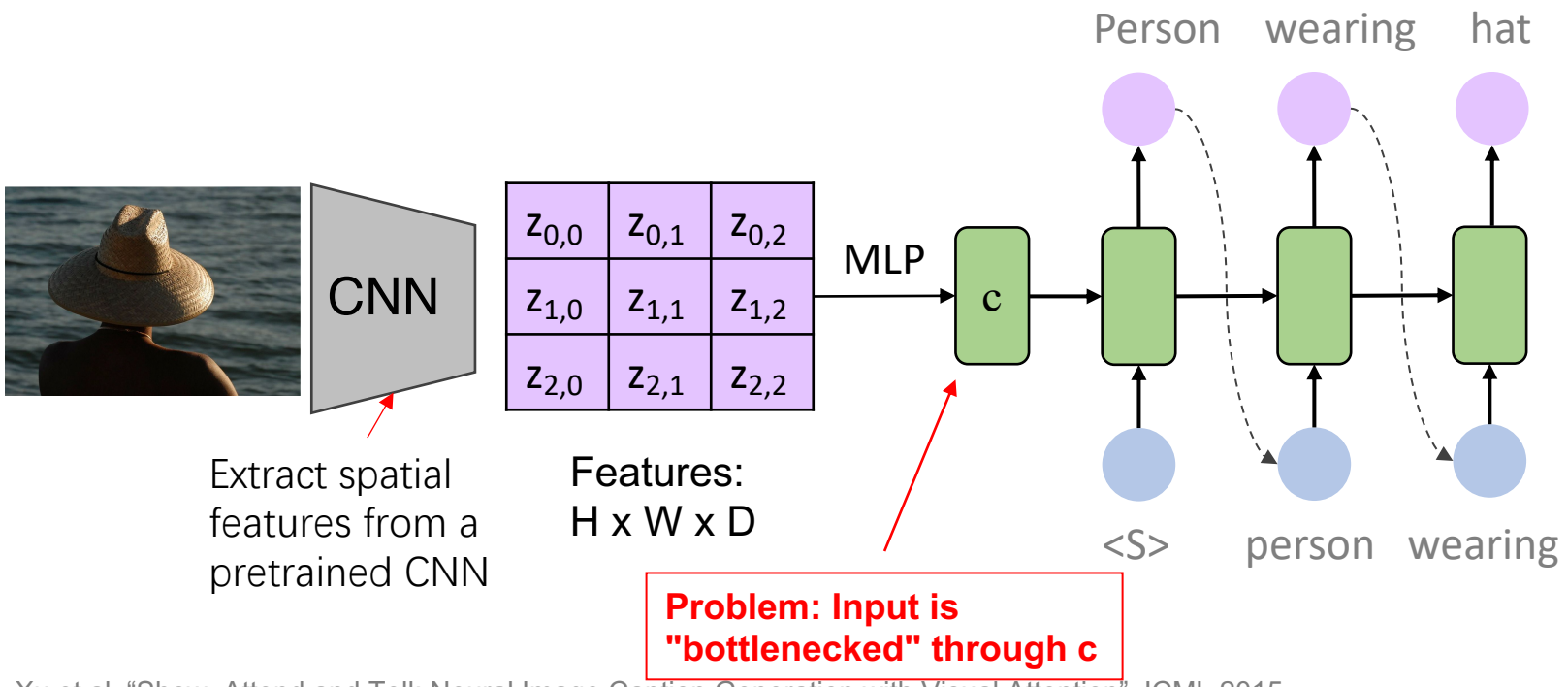
Input: Image **I**

Output: Sequence $\mathbf{y} = y_1, y_2, \ldots, y_T$

Idea: Encoder-Decoder?

Encoder: $c = f_W(\mathbf{z})$
where $\mathbf{z}$ is spatial CNN features
$f_W(.)$ is an MLP

Decoder: $y_t = g_V(y_{t-1}, h_{t-1}, c)$
where context vector c is often $h_0 = c$



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Encoder-Decoder with Attention?

- Each context vector will attend to different image regions

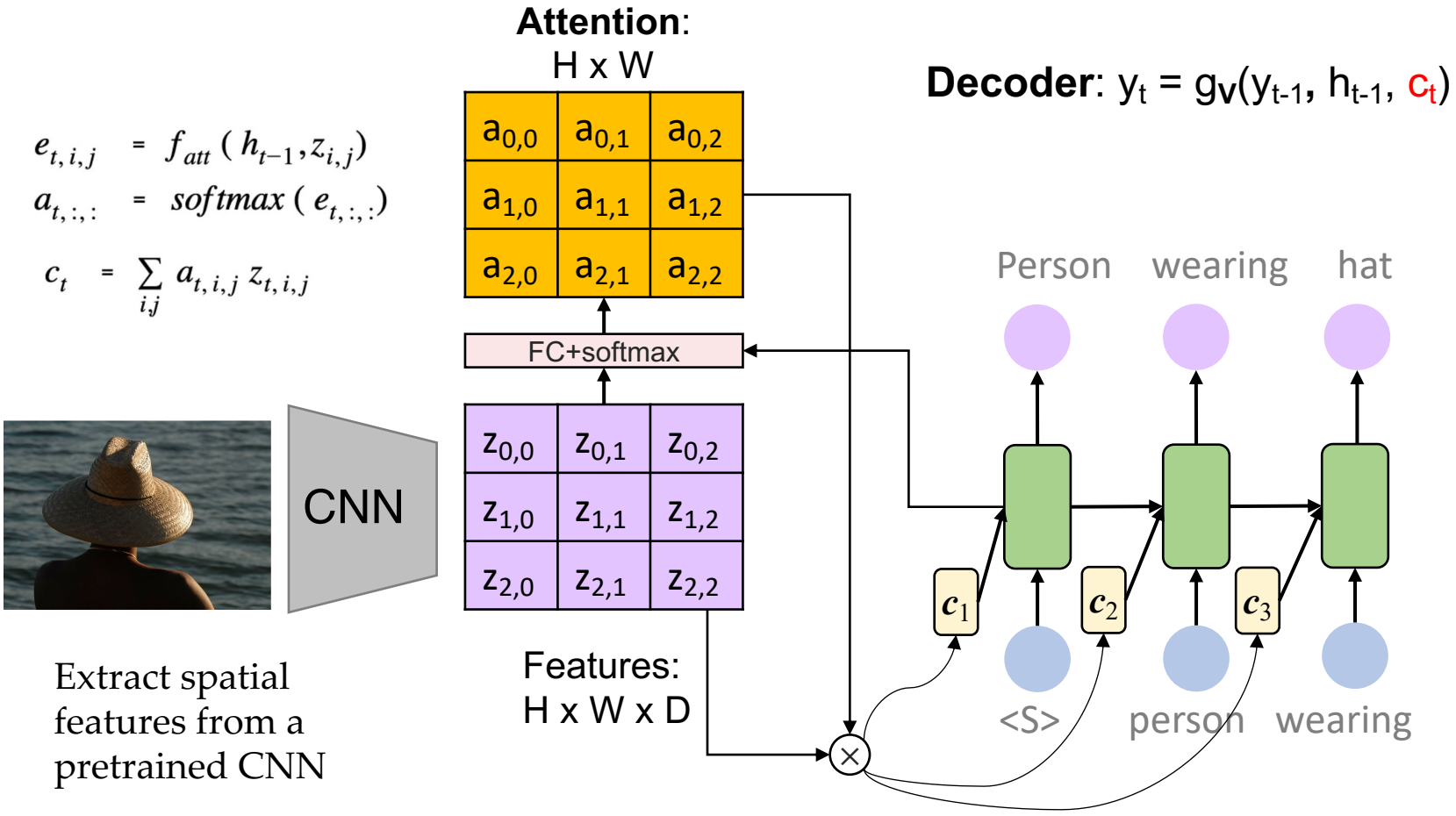


Image Captioning with Attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu et al, “Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention”, ICML 2015

CNN with Self-Attention

- We can also apply **self-attention** to CNN features.

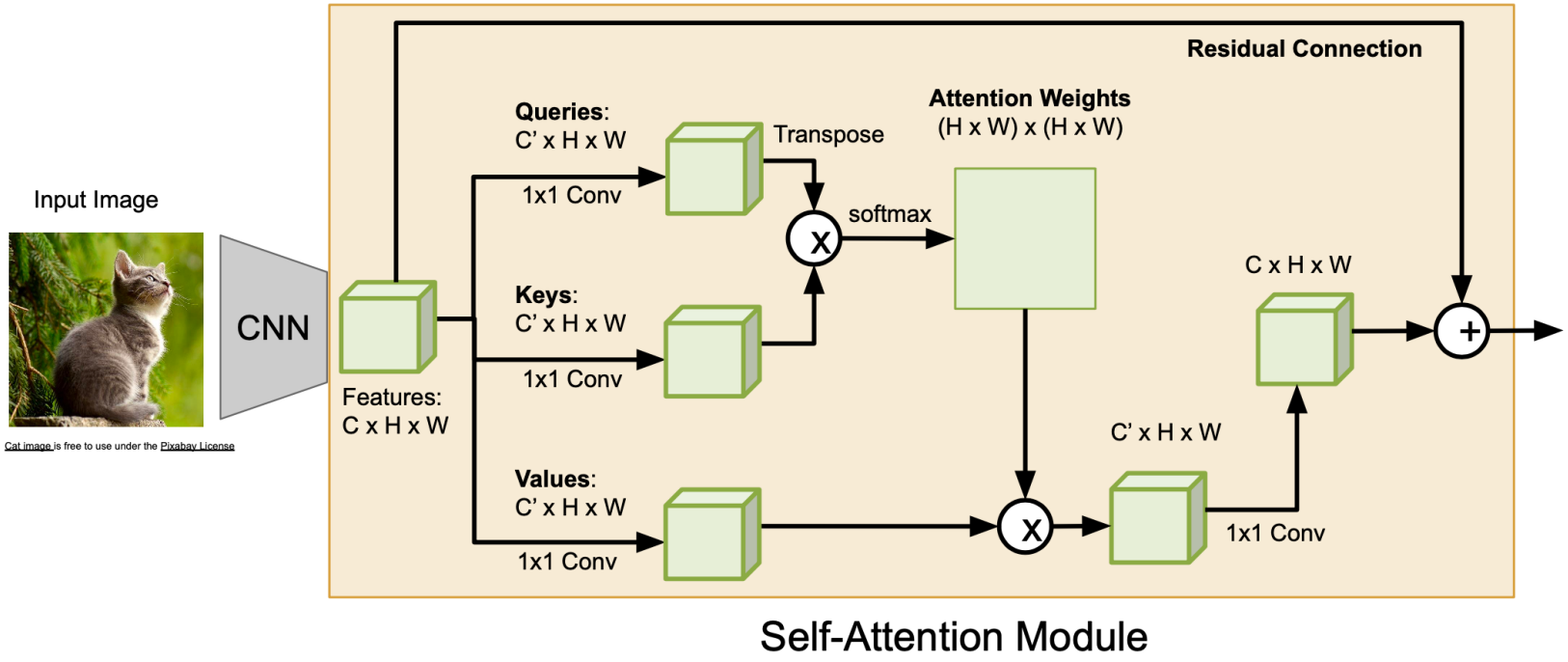
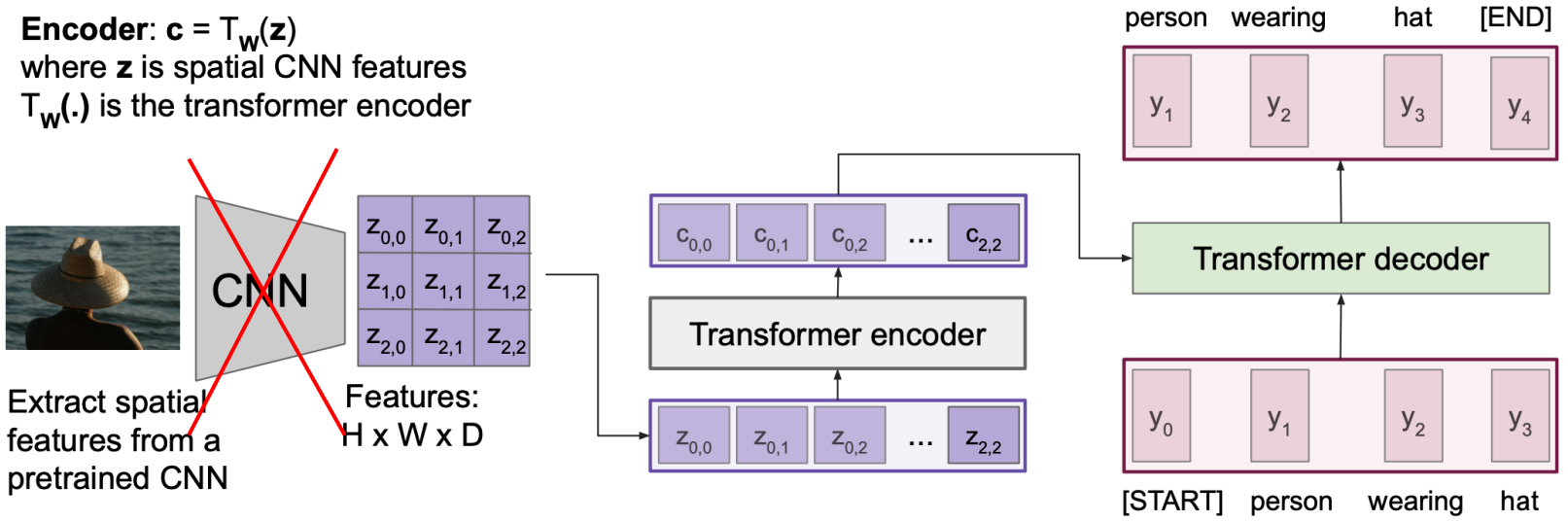
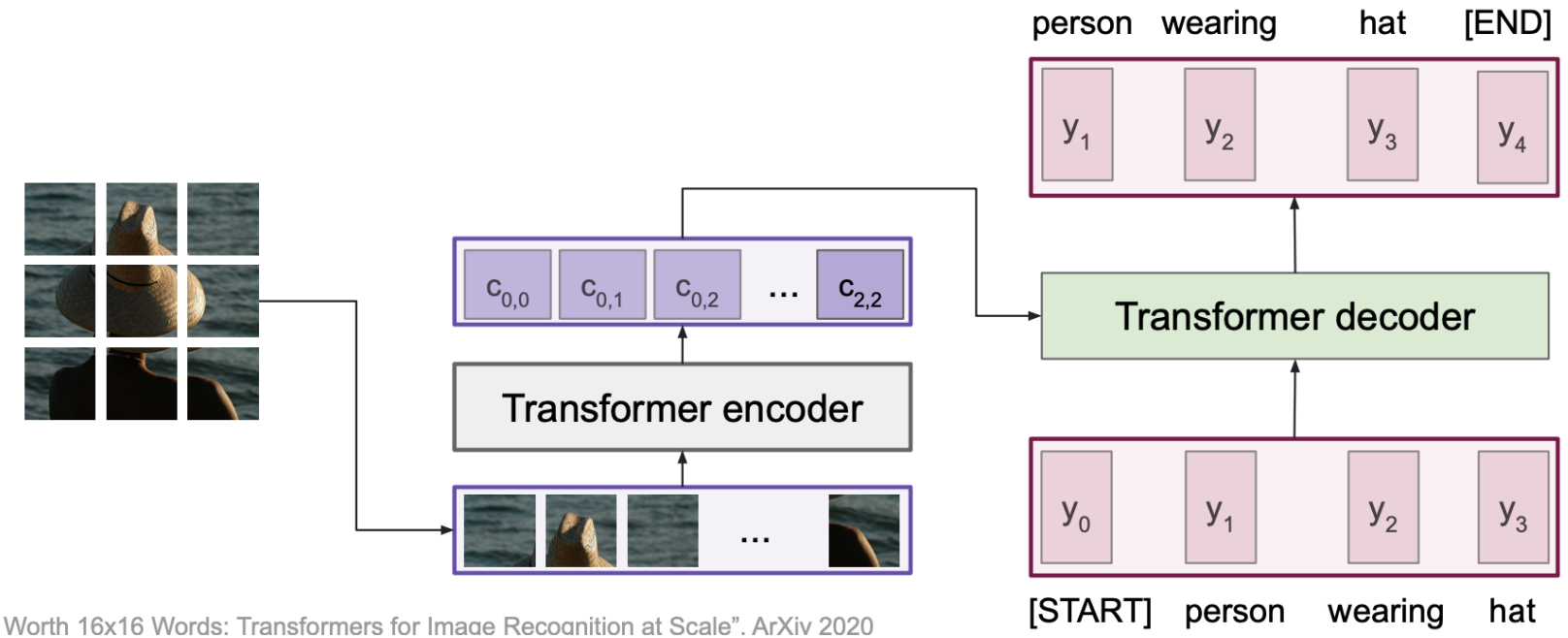


Image Captioning ~~with~~using Transformers



Perhaps we don't need convolutions at all?

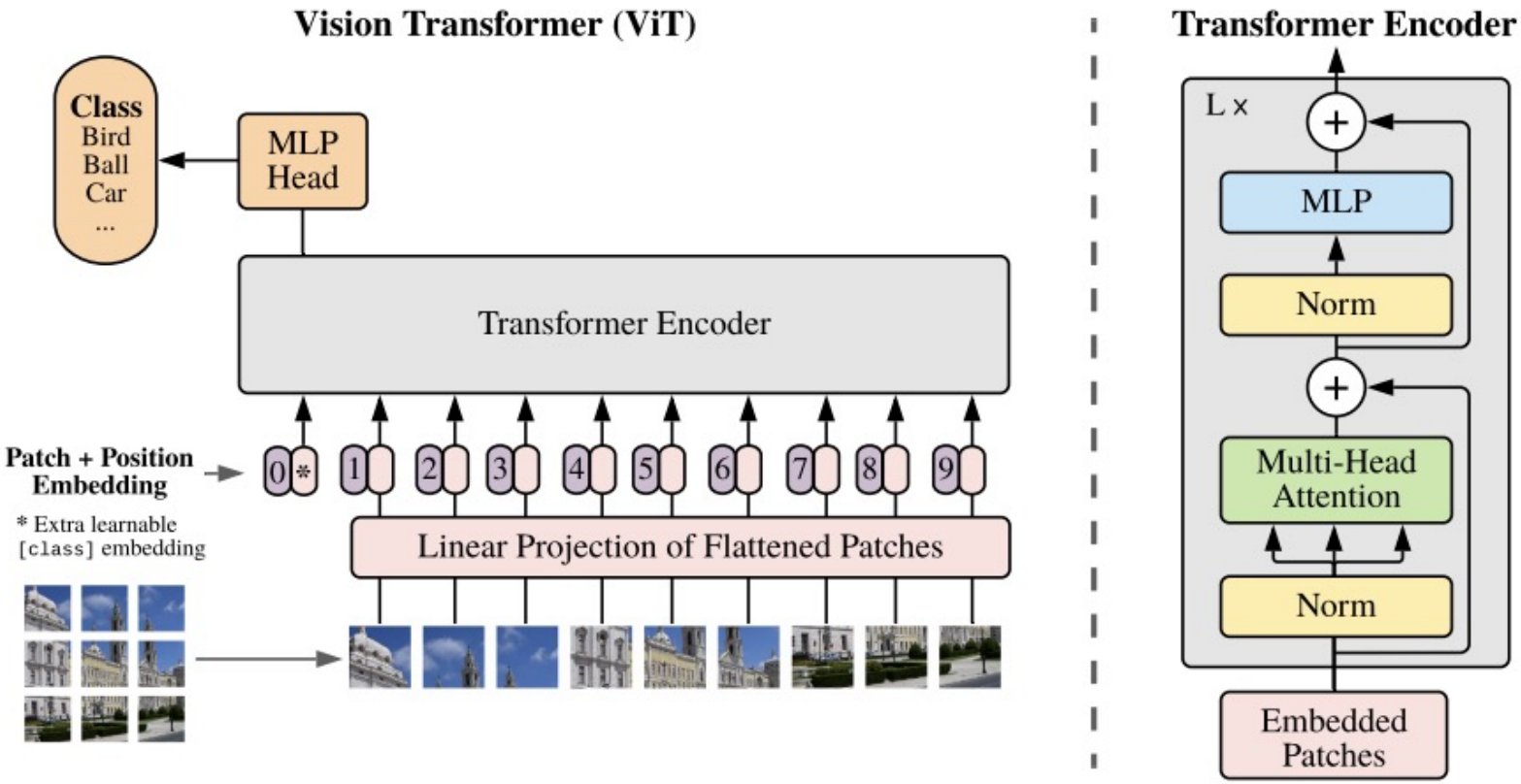
Image Captioning using ONLY Transformers



Worth 16x16 Words: Transformers for Image Recognition at Scale". ArXiv 2020

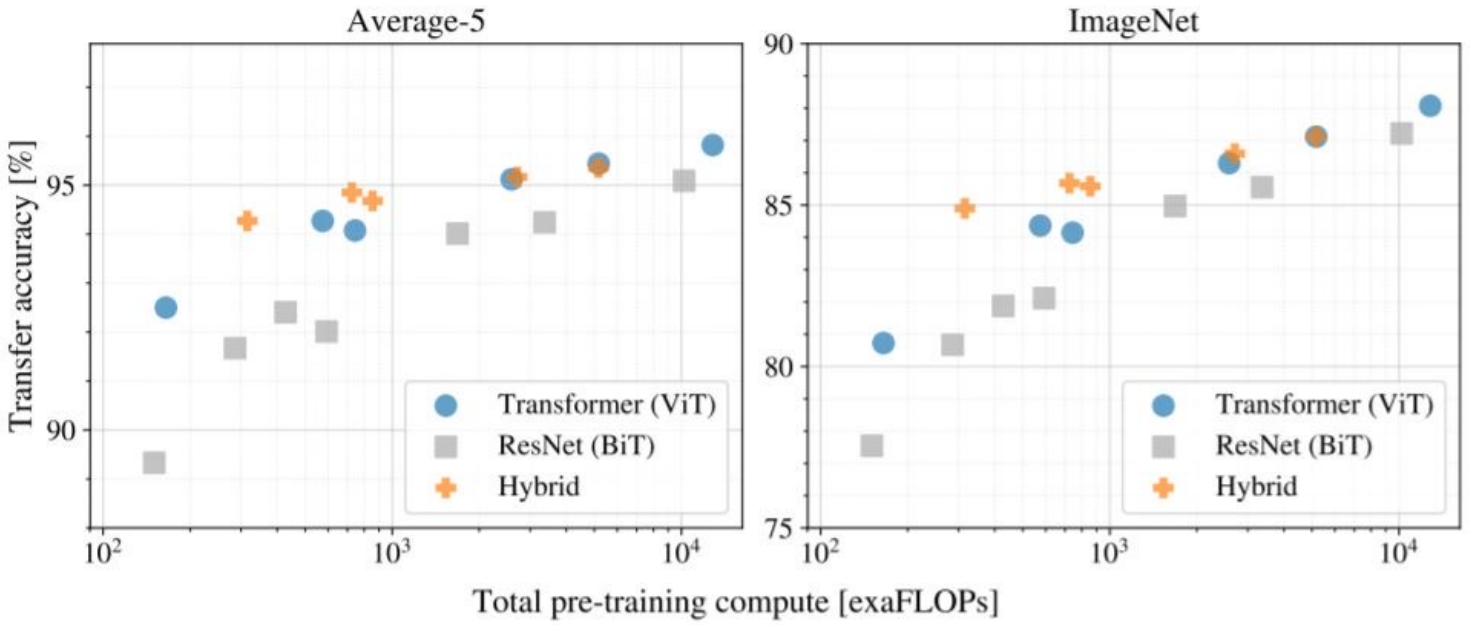
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ArXiv 2020

Vision Transformer (ViT)



Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ArXiv 2020

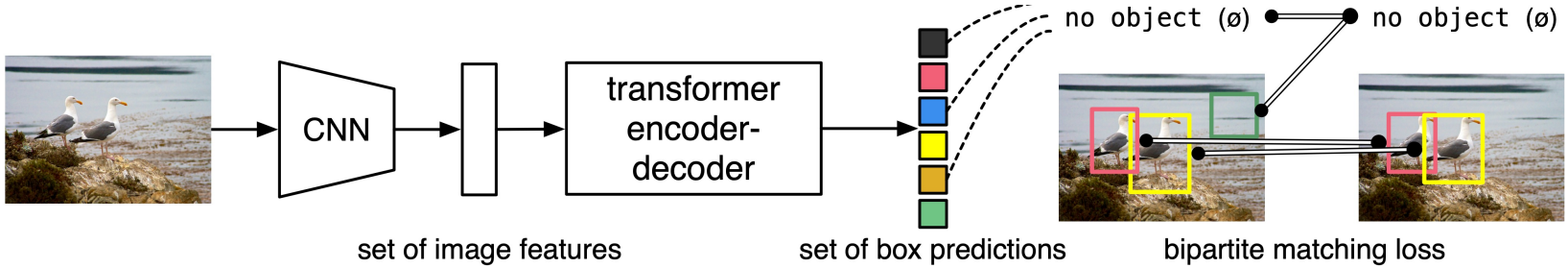
Vision Transformers vs. ResNets

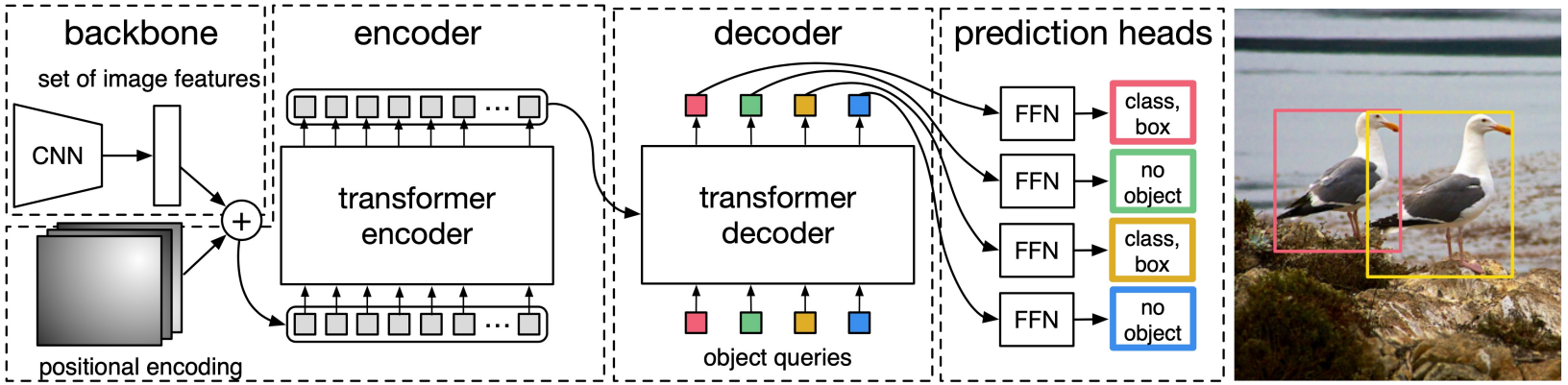


Farewell CNN ?

Vision Transformers for Object Detection

- DE:TR

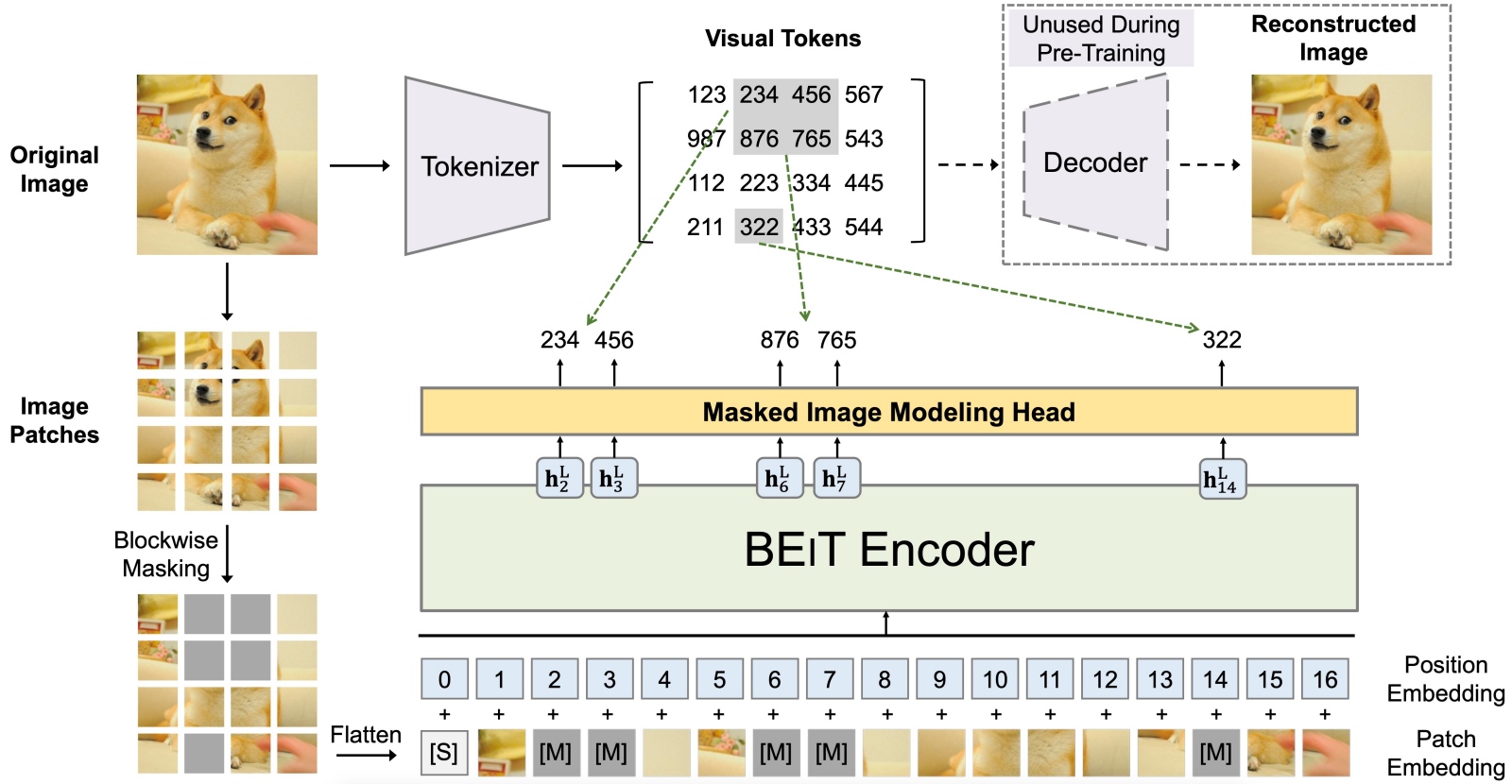




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

BEIT: Pre-Trained Models for Images

- Backbone Network: Standard Transformer
- Pre-Training: Masked Image Modeling



BEIT: BERT Pre-Training of Image Transformers

BEIT

- State-of-the-art performance on CV tasks

Models	Model Size	Resolution	ImageNet
Training from scratch (i.e., random initialization)			
ViT_{384}-B [DBK+20]	86M	384^2	77.9
ViT_{384}-L [DBK+20]	307M	384^2	76.5
DeiT-B [TCD+20]	86M	224^2	81.8
$DeiT_{384}$-B [TCD+20]	86M	384^2	83.1
Supervised Pre-Training on ImageNet-22K (using labeled data)			
ViT_{384}-B [DBK+20]	86M	384^2	84.0
ViT_{384}-L [DBK+20]	307M	384^2	85.2
Self-Supervised Pre-Training on ImageNet-1K (without labeled data)			
iGPT-1.36B† [CRC+20]	1.36B	224^2	66.5
ViT_{384}-B-JFT300M‡ [DBK+20]	86M	384^2	79.9
MoCo v3-B [CXH21]	86M	224^2	83.2
MoCo v3-L [CXH21]	307M	224^2	84.1
DINO-B [CTM+21]	86M	224^2	82.8
BEIT-B (ours)	86M	224^2	83.2
$BEIT_{384}$-B (ours)	86M	384^2	84.6
BEIT-L (ours)	307M	224^2	85.2
$BEIT_{384}$-L (ours)	307M	384^2	**86.3**

Top-1 accuracy on ImageNet-1K

ADE20K semantic segmentation

Models	mIoU (%)	Multi-Scale mIoU (%)
Supervised Pre-Training on ImageNet-22K (using labeled data)		
Swin-B [LLC+21]	50.0	51.7
Swin-L [LLC+21]	52.1	53.5
Self-Supervised Pre-Training, and Intermediate Fine-Tuning on ImageNet-22K		
BEIT-B^+ (ours)	53.6	54.2
BEIT-L^+ (ours)	56.7	57.0
Self-Supervised Pre-Training, and Intermediate Fine-Tuning on In-House-70M		
BEIT-L^+ (ours)	**57.9**	**58.4**

What's Next?

- **Unsupervised Learning**

Clustering

Learn hidden structures

- K-means
- Gaussion Mixture

Generative

density estimation

- GAN
- VAE
- Diffusion

Dimensional Reduction

Learning data compression

- Principal component analysis
- Auto-encoder