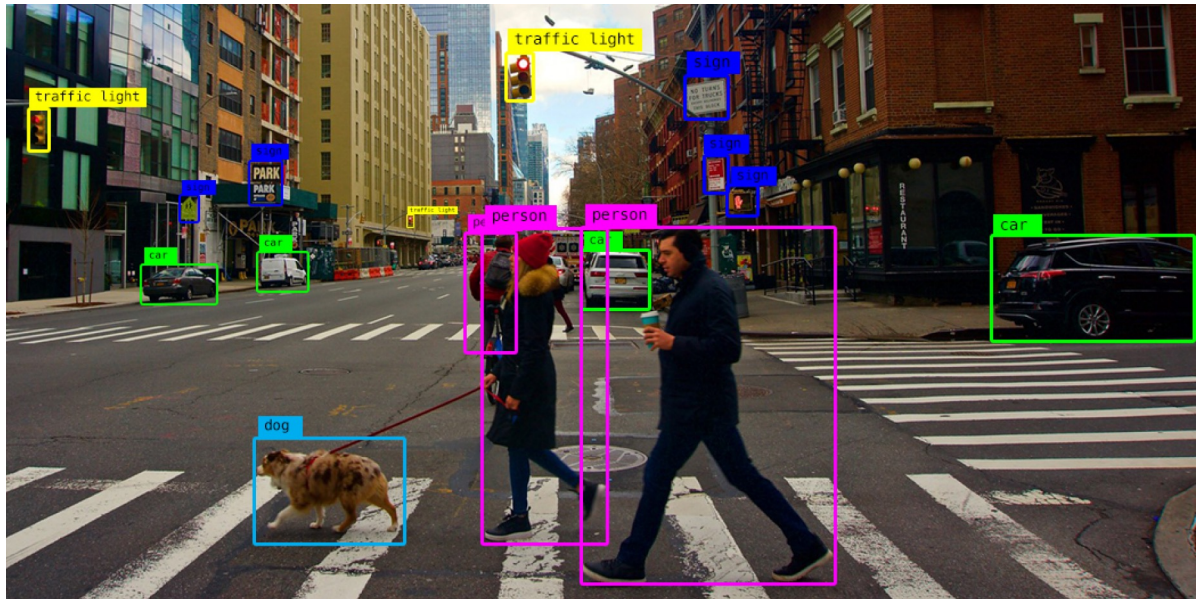
AI VIETNAM
All-in-One Course

Object Detection
(Part 2: CNN-based Algorithms)



Vinh Dinh Nguyen - PhD in Computer Science
Hung-An & Minh-Duc Bui - TA

Year 2023

Outline

- CNN Limitations
- Region Based Convolutional Neural Networks
- Spatial Pyramid Pooling
- Fast R-CNN
- Faster RCNN
- Object Detection with Transformers: Motivation

Outline

- CNN Limitations
- Region Based Convolutional Neural Networks
- Spatial Pyramid Pooling
- Fast R-CNN
- Faster RCNN
- Object Detection with Transformers: Motivation

Object Detection Milestones

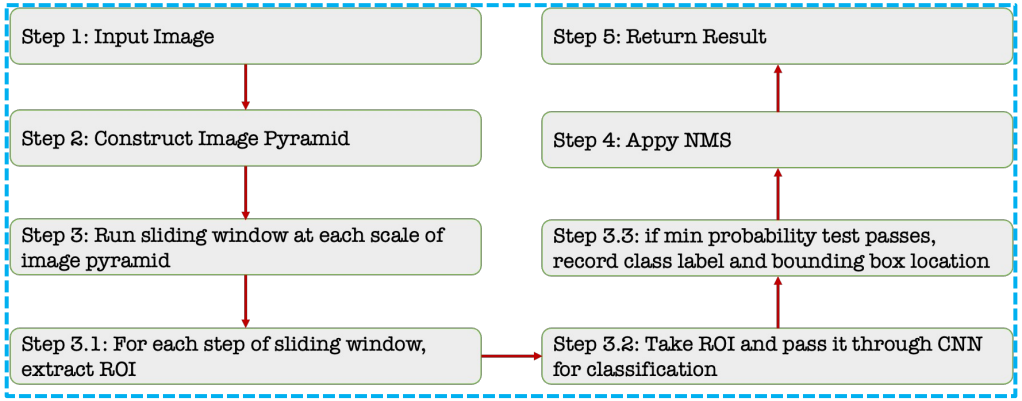
Milestones: Traditional Detectors
Viola Jones Detectors, SVM + HOG & DPM

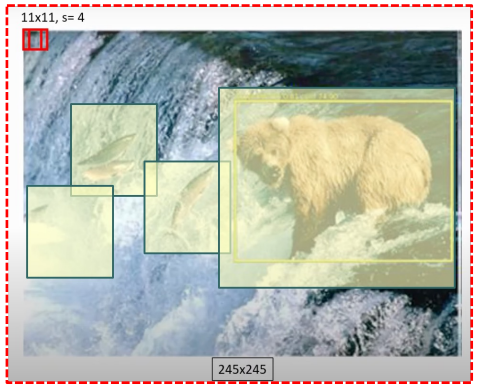
Milestones: CNN based Two-stage Detectors
RCNN, SPPNet, Faster RCNN, Faster RCNN,..

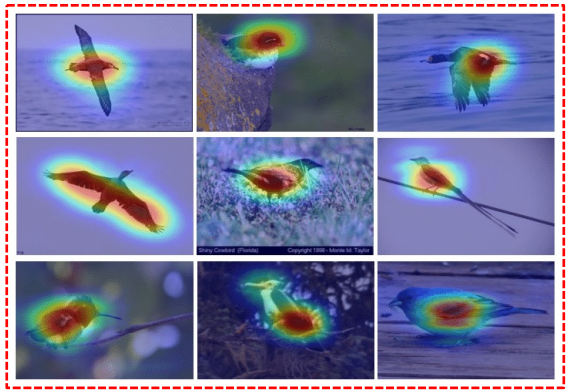
Milestones: CNN based One-stage Detectors
YOLO, SSD, RetinaNet, CornerNet, Center Net,..

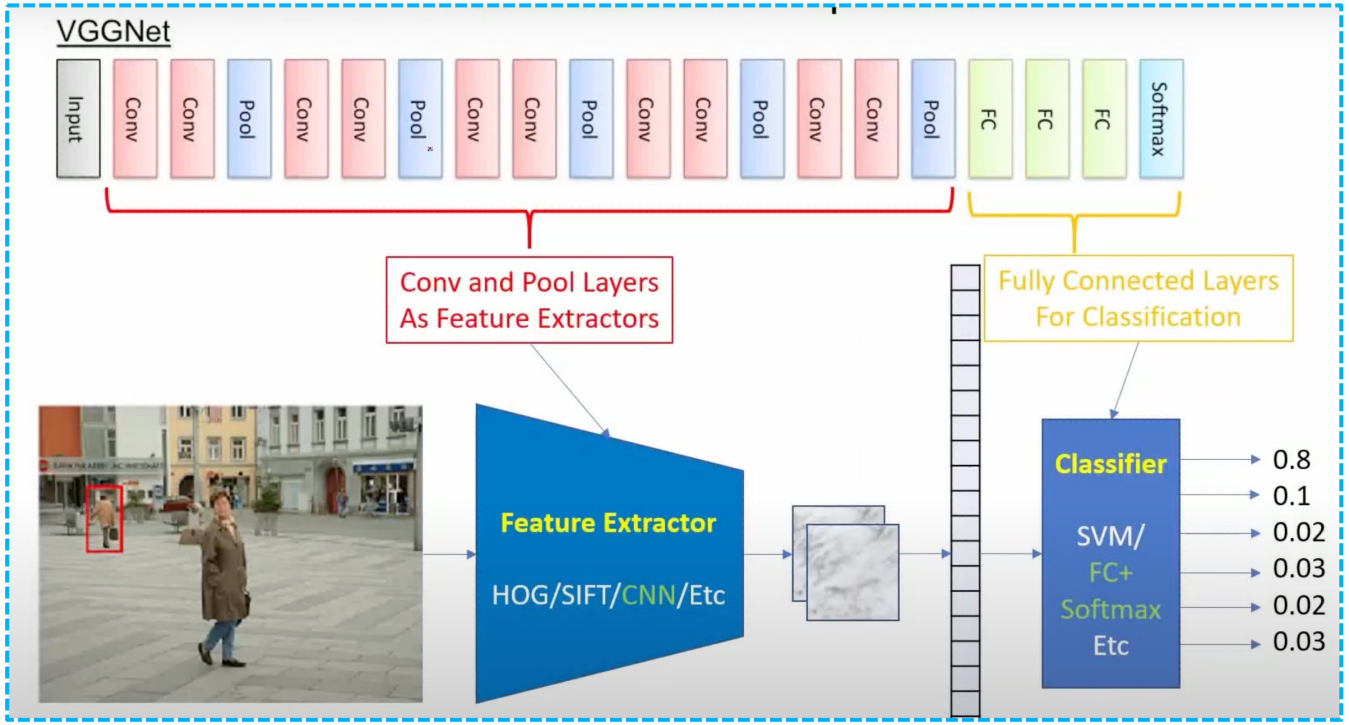
Milestones: Transformer for OD
DETR, D-DETR, DINO,...

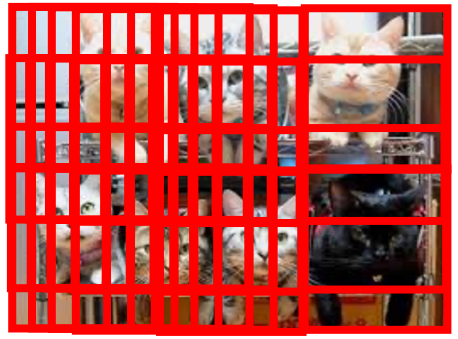
Convolutional Neural Network's Limitations







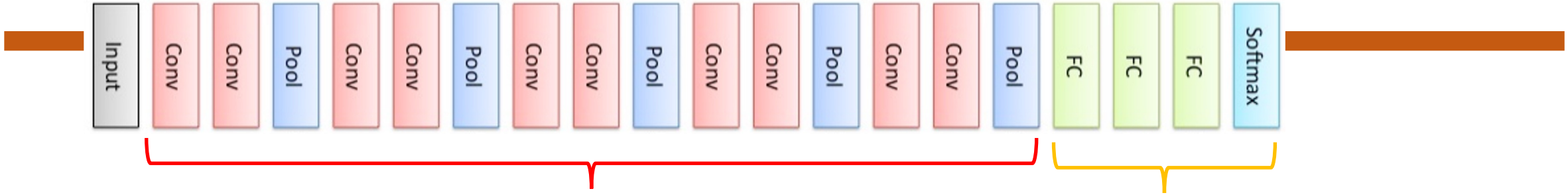




We need to focus on object only, not entire image

Classification Pipeline

All VGGNet



Conv and Pool Layers
As Feature Extractors

Fully Connected Layers
For Classification

Feature Extractor

HOG/SIFT/CNN/Etc

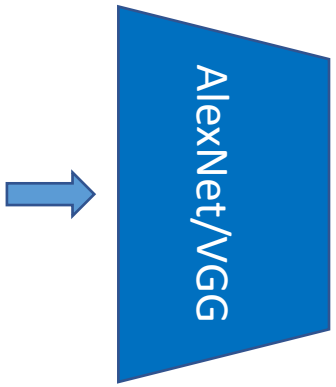
Classifier

SVM/
FC+
Softmax
Etc

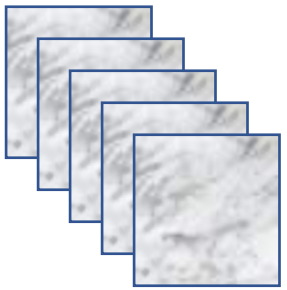
0.8
0.1
0.02
0.03
0.02
0.03

Classification

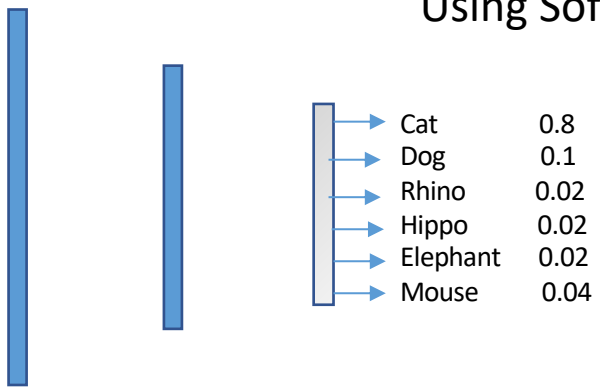
CAT



Conv and Pool Layers
As Feature Extractors



Feature Maps



Fully Connected Layers
For Classification

Image Credit - http://host.robots.ox.ac.uk/pascal/VOC/voc2012/examples/index.html

Ideas for *Localization* using ConvNets

Case #1 – Only one object per image

AlexNet/VGG

10
Human
Car
Dog
Cat
Bicycle
etc

Get Class Scores
Using Softmax

X1, y1
X2, y2
w
h
X0, y0

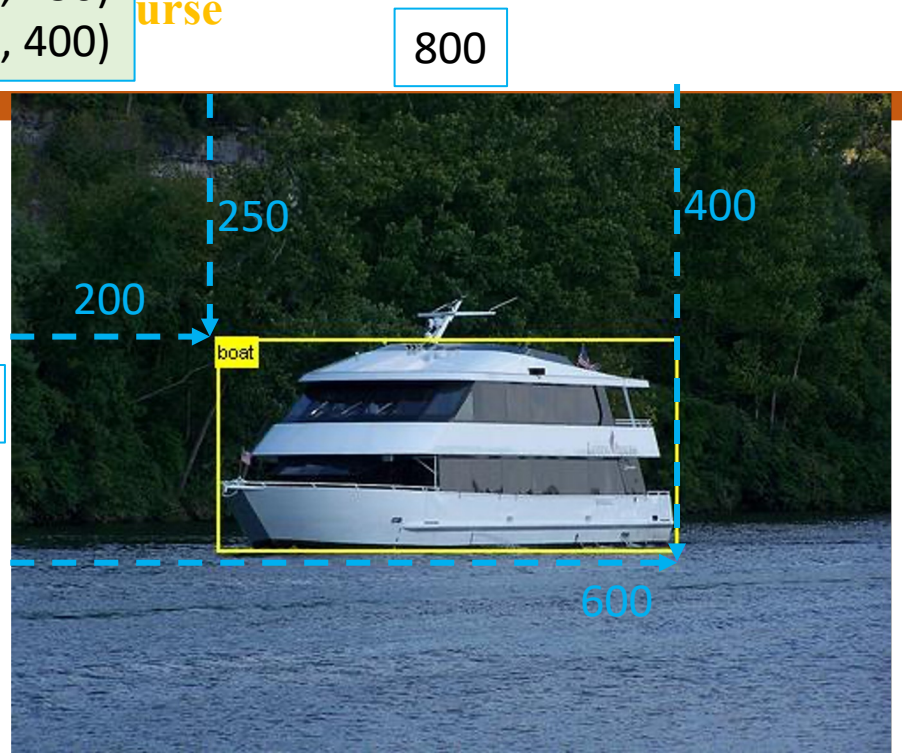
Get Bounding boxes
Using L2 loss
(x1, y1, x2, y2)

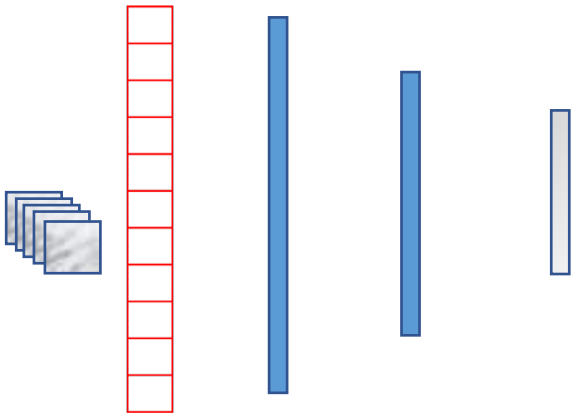
Human
Car
Dog
Cat
Bicycle
etc

Image Credit - http://host.robots.ox.ac.uk/pascal/VOC/voc2012/examples/index.html

Bounding Box Regression Training

(x1,y1) = (200, 250)
(x2,y2) = (600, 400)





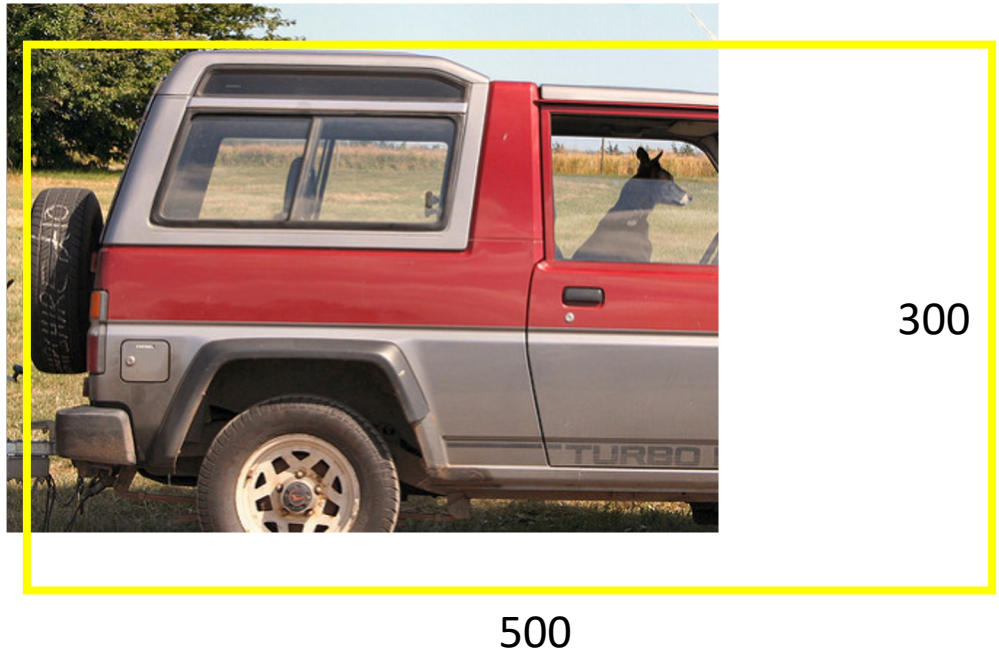
	x1	y1	x2	y2	L2 Loss				
Expected	200	250	600	400					
Prediction	0	0	800	600	$(200-0)^2$	$(250-0)^2$	$(600-800)^2$	$(400-600)^2$	182500
	100	150	700	450	$(200-100)^2$	$(250-150)^2$	$(600-700)^2$	$(400-450)^2$	32500
	210	245	590	405	$(200-210)^2$	$(250-245)^2$	$(600-590)^2$	$(400-405)^2$	250
	200	250	600	400	$(200-200)^2$	$(250-250)^2$	$(600-600)^2$	$(400-400)^2$	0

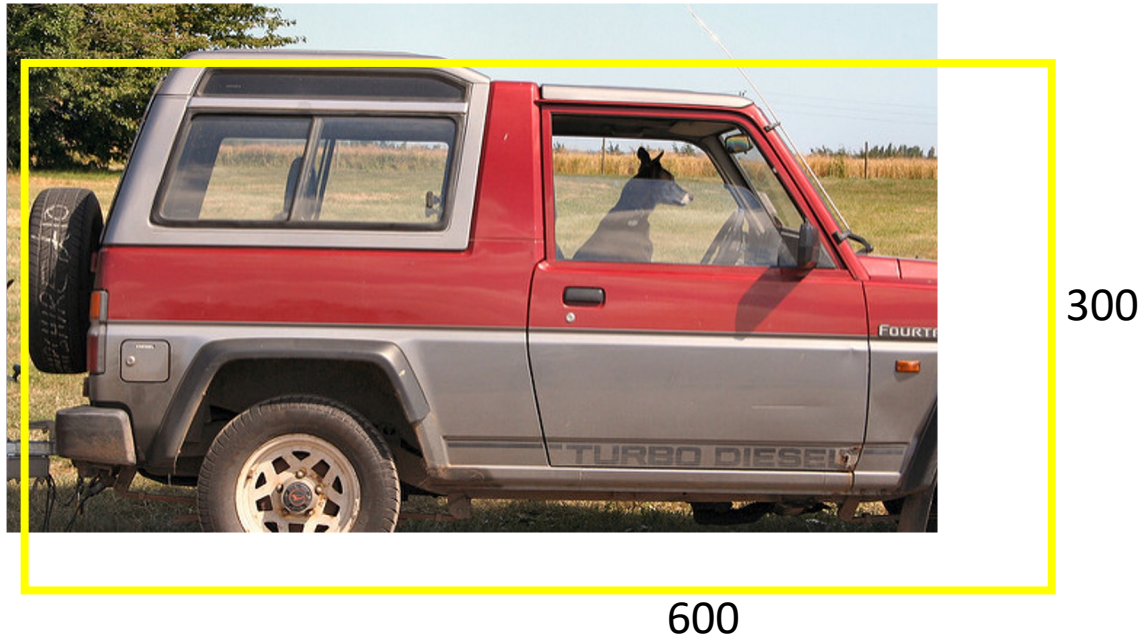
Image Credit - http://host.robots.ox.ac.uk/pascal/VOC/voc2012/examples/index.html

BBox General Properties

Image Credits - see Notes

About Bounding Boxes





Ideas for *Localization* using ConvNets

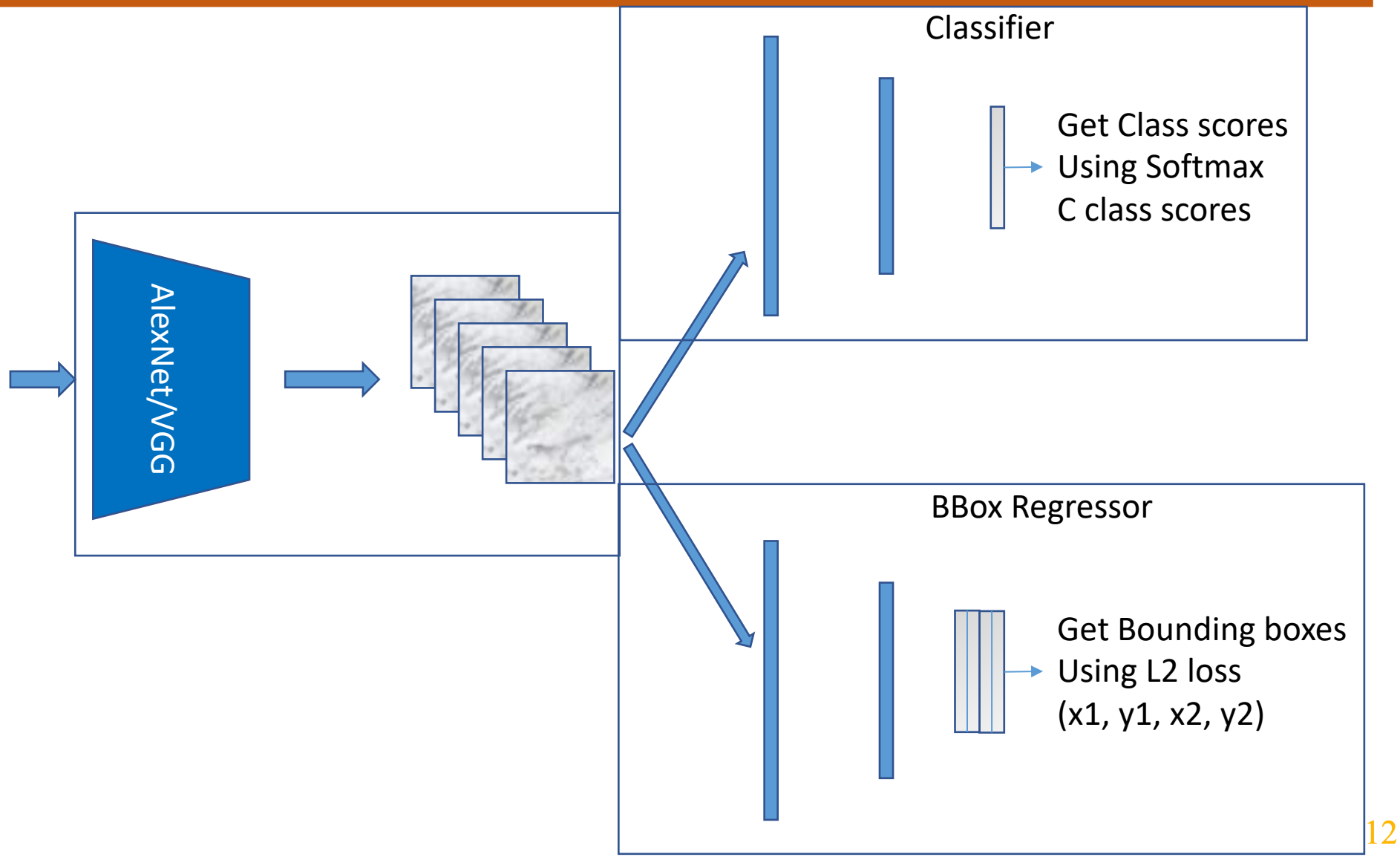
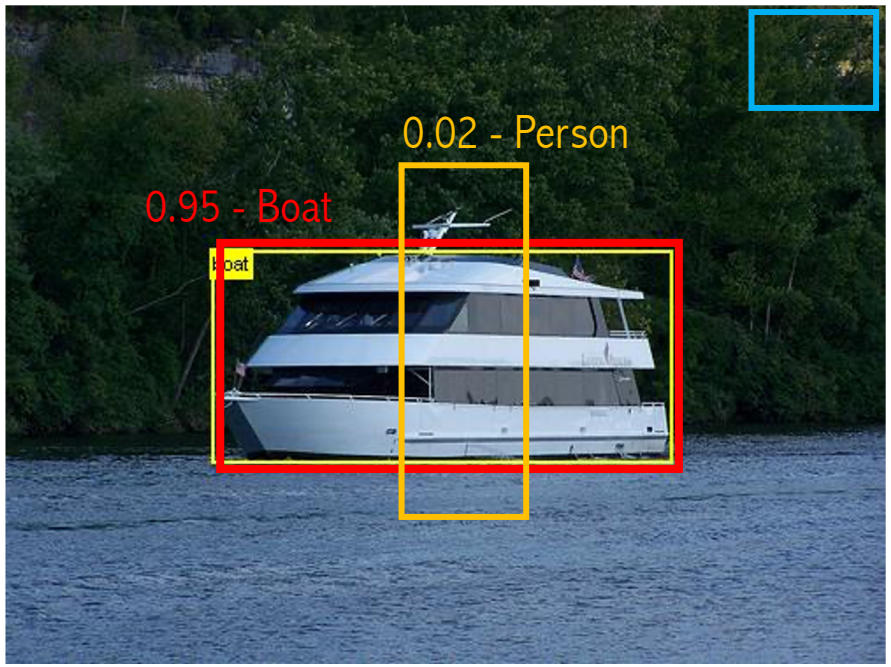
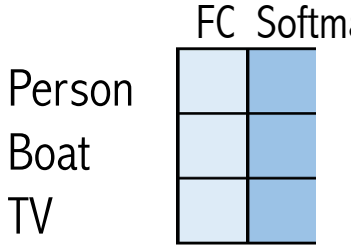


Image Credit - http://host.robots.ox.ac.uk/pascal/VOC/voc2012/examples/index.html

Combining Results





Class	Conf	Bbox coordinates			
Person	0.02	380	200	430	400
Boat	0.95	210	245	590	405
TV	0.03	700	10	790	100

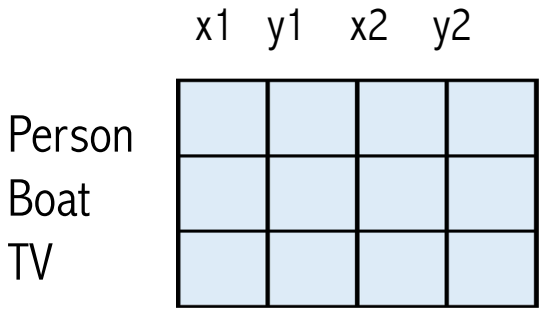
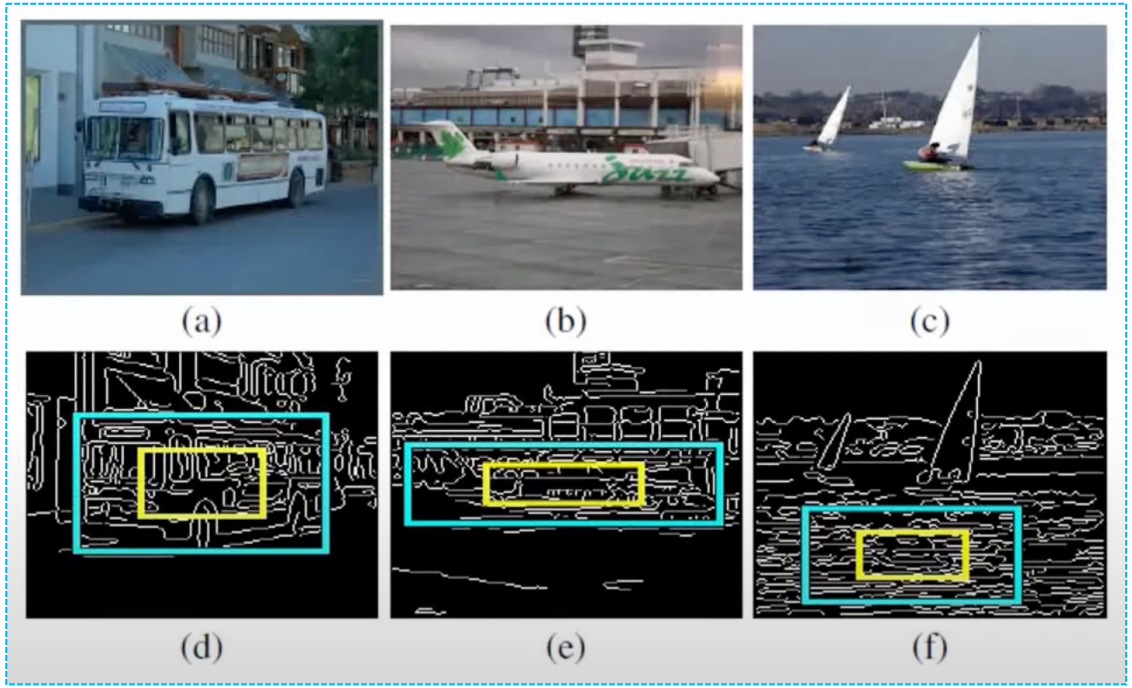


Image Credit - http://host.robots.ox.ac.uk/pascal/VOC/voc2012/examples/index.html

Region Proposal Methods



Edge Density

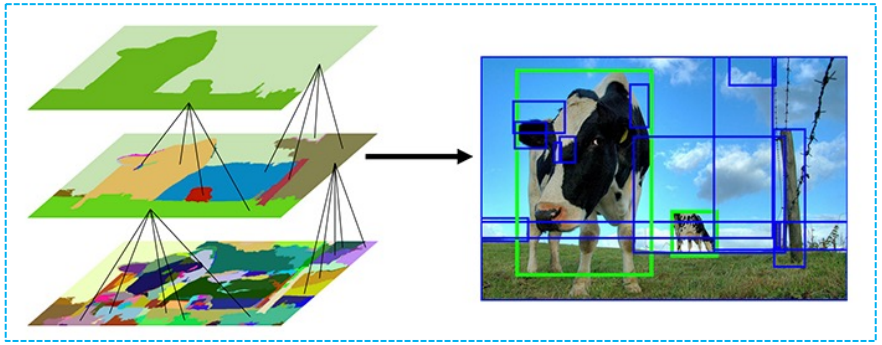
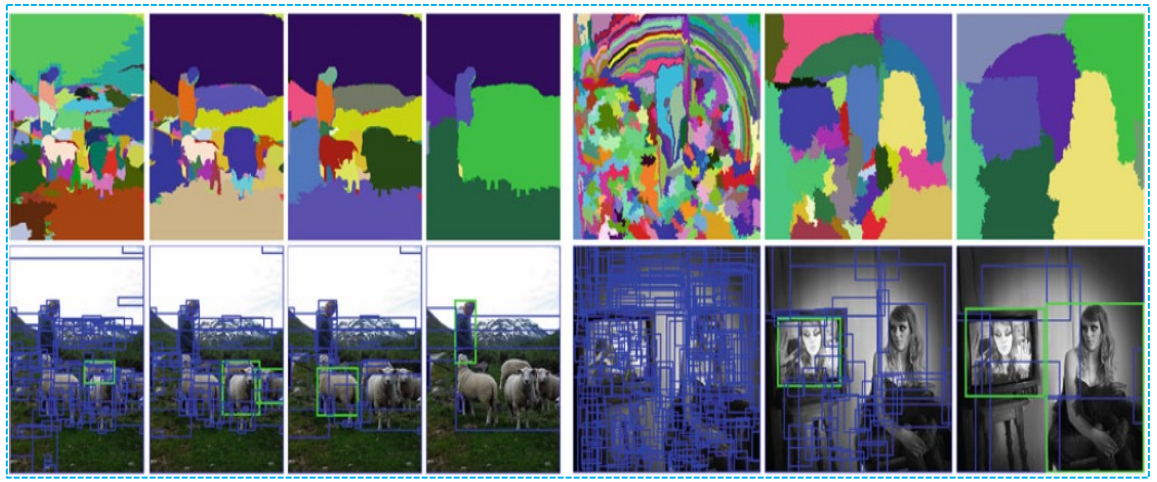


Image Segmentation

Method		Time	Repeatability	Recall	Detection	mAP
Objectness[1]	O	3	·	★	·	25.0/25.4
CPMC[4]	C	250	-	★★	★	29.9/30.7
Endres2010[9]	E	100	-	★★	★★	31.2/31.6
Sel.Search[30]	SS	10	★★	★★★	★★	31.7/32.3
Rahtu2011[24]	R1	3	·	·	★	29.6/30.4
Rand.Prim[22]	RP	1	★	★	★	30.5/30.9
Bing[6]	B	0.2	★★★	★	·	21.8/22.4
MCG[3]	M	30	★	★★★	★★	**32.4/32.7**
Ranta.2014[25]	R4	10	★★	·	★	30.7/31.3
EdgeBoxes[33]	EB	0.3	★★	★★★	★★	31.8/32.2
Uniform	U	0	·	·	·	16.6/16.9
Gaussian	G	0	·	·	★	**27.3/28.0**
SlidingWindow	SW	0	★★★	·	·	20.7/21.5
Superpixels	SP	1	★	·	·	11.2/11.3

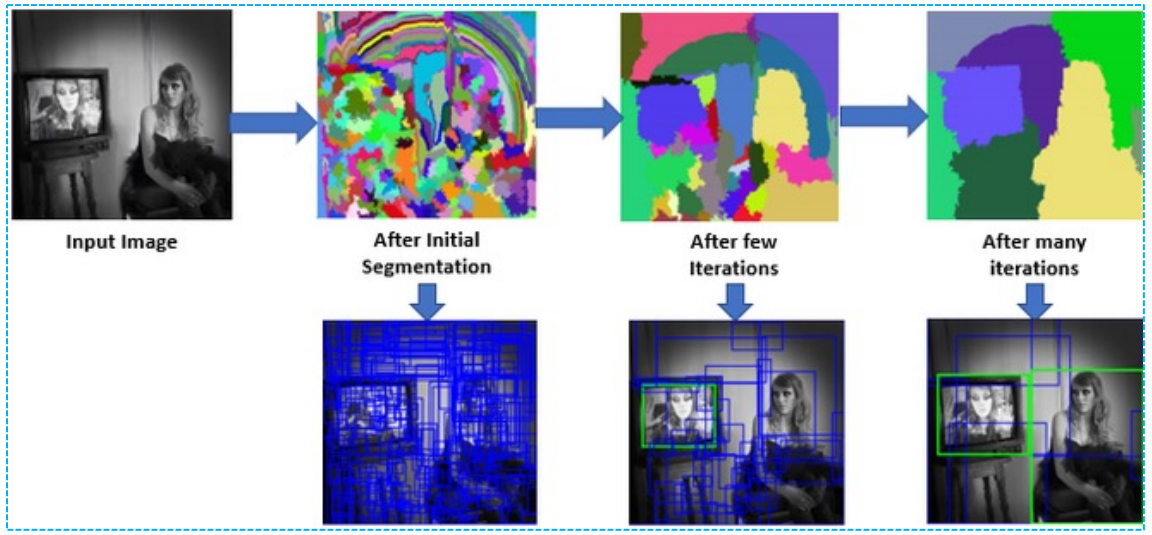
Jan Hosang, Rodrigo Benenson, Bernt Schiele, "How good are detection proposals, really?", 2014

Edge Boxes & Selective Search



Zitnick, C.L., Dollár, P. (2014). Edge Boxes: Locating Object Proposals from Edges.

How to Apply Edge Boxes and Selective Search for CNN?



Region Proposal Algorithms

Region proposal algorithms identify prospective objects in an image using segmentation. In segmentation, we group adjacent regions which are similar to each other based on some criteria such as color, texture etc.

RBG Image

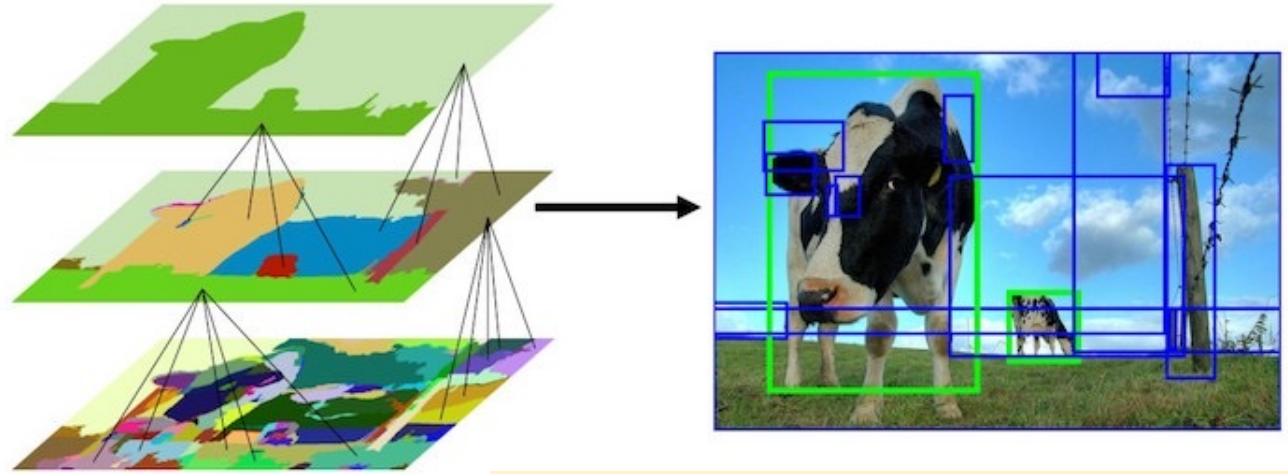
Segmentation Result

Can we use segmented parts in this image as region proposals? The answer is no and there are two reasons why we cannot do that

1.Most of the actual objects in the original image contain 2 or more segmented parts
2.Region proposals for occluded objects such as the plate covered by the cup or the cup filled with coffee cannot be generated using this method

Selective Search

Selective search uses oversegments from Felzenszwalb and Huttenlocher's method as an initial seed. An oversegmented image looks like this.



Selective Search uses 4 similarity measures based on color, texture, size and shape compatibility.

1. Add all bounding boxes corresponding to segmented parts to the list of regional proposals
2. Group adjacent segments based on similarity
3. Go to step 1

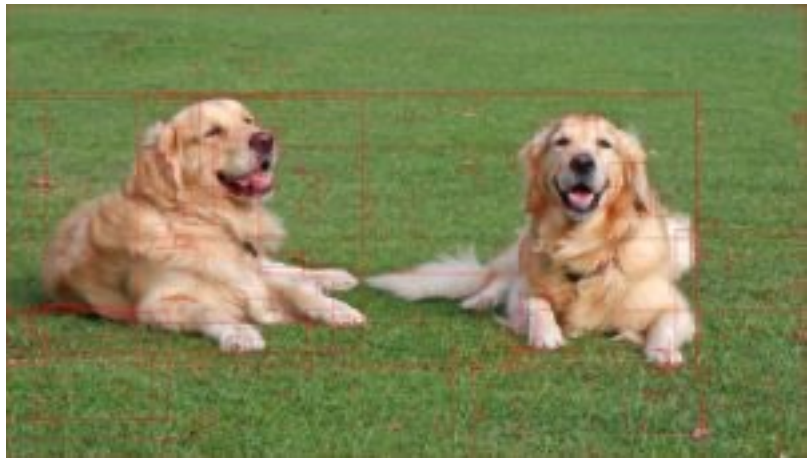
$$s_{color}(r_i, r_j) = \sum_{k=1}^{n} min(c_i^k, c_j^k) \quad s_{texture}(r_i, r_j) = \sum_{k=1}^{n} min(t_i^k, t_j^k)$$

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)} \quad s_{fill}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)}$$

$$s(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j)$$

Selective Search

Selective search uses oversegments from Felzenszwalb and Huttenlocher's method as an initial seed. An oversegmented image looks like this.



Dogs: top 250 region proposals

Breakfast Table: top 200 region proposals

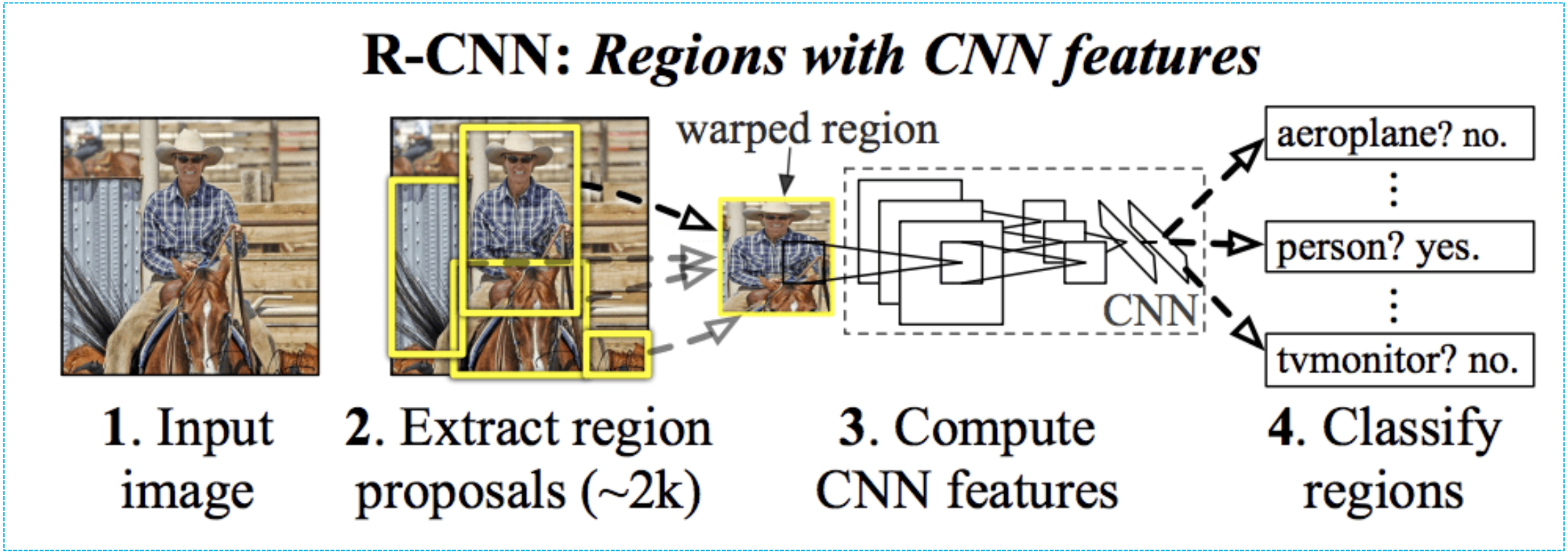
1. Add all bounding boxes corresponding to segmented parts to the list of regional proposals
2. Group adjacent segments based on similarity
3. Go to step 1

https://learnopencv.com/selective-search-for-object-detection-cpp-python/

Outline

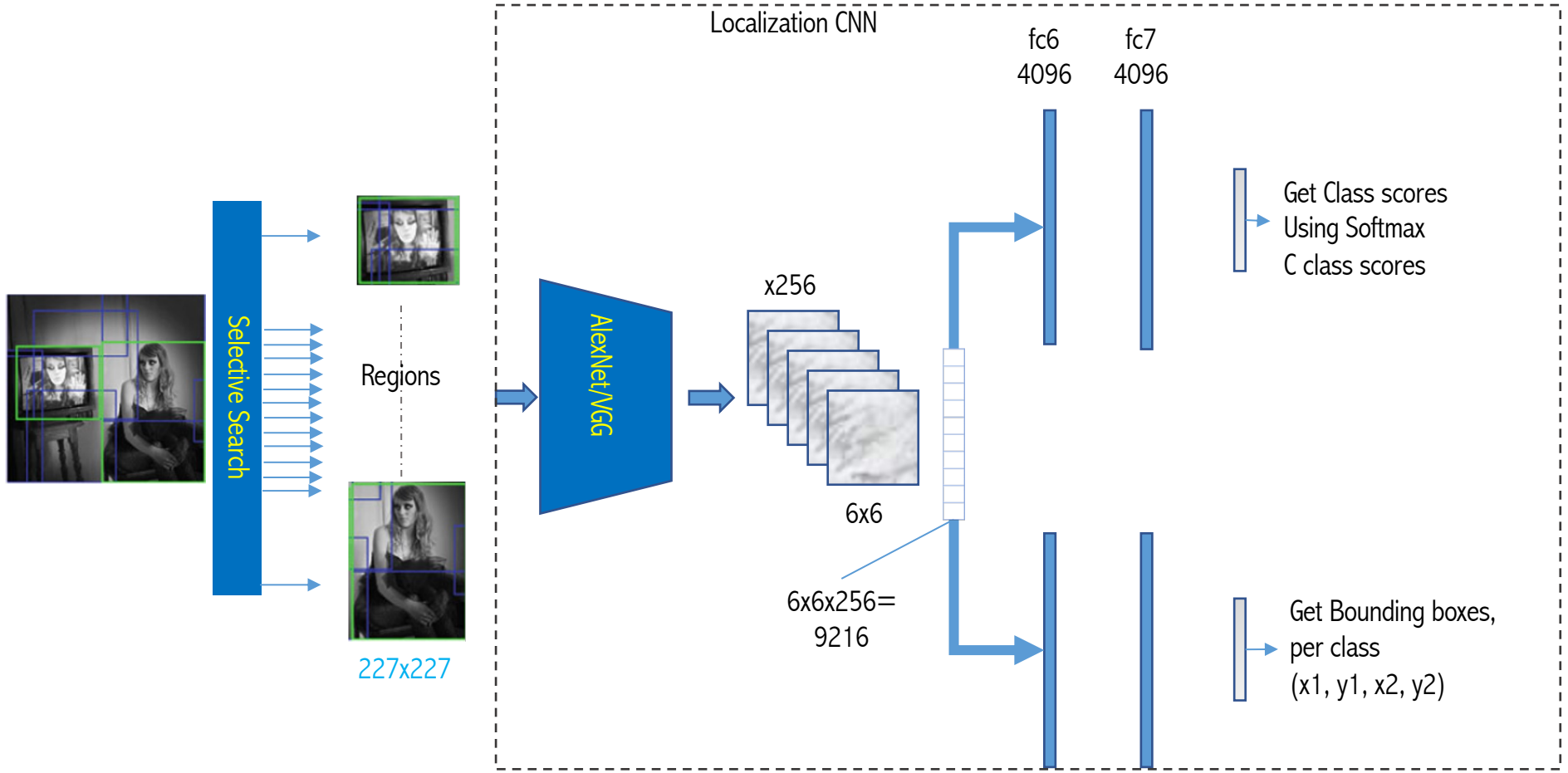
- CNN Limitations
- Region Based Convolutional Neural Networks
- Spatial Pyramid Pooling
- Fast R-CNN
- Faster RCNN
- Object Detection with Transformers: Motivation

R-CNN



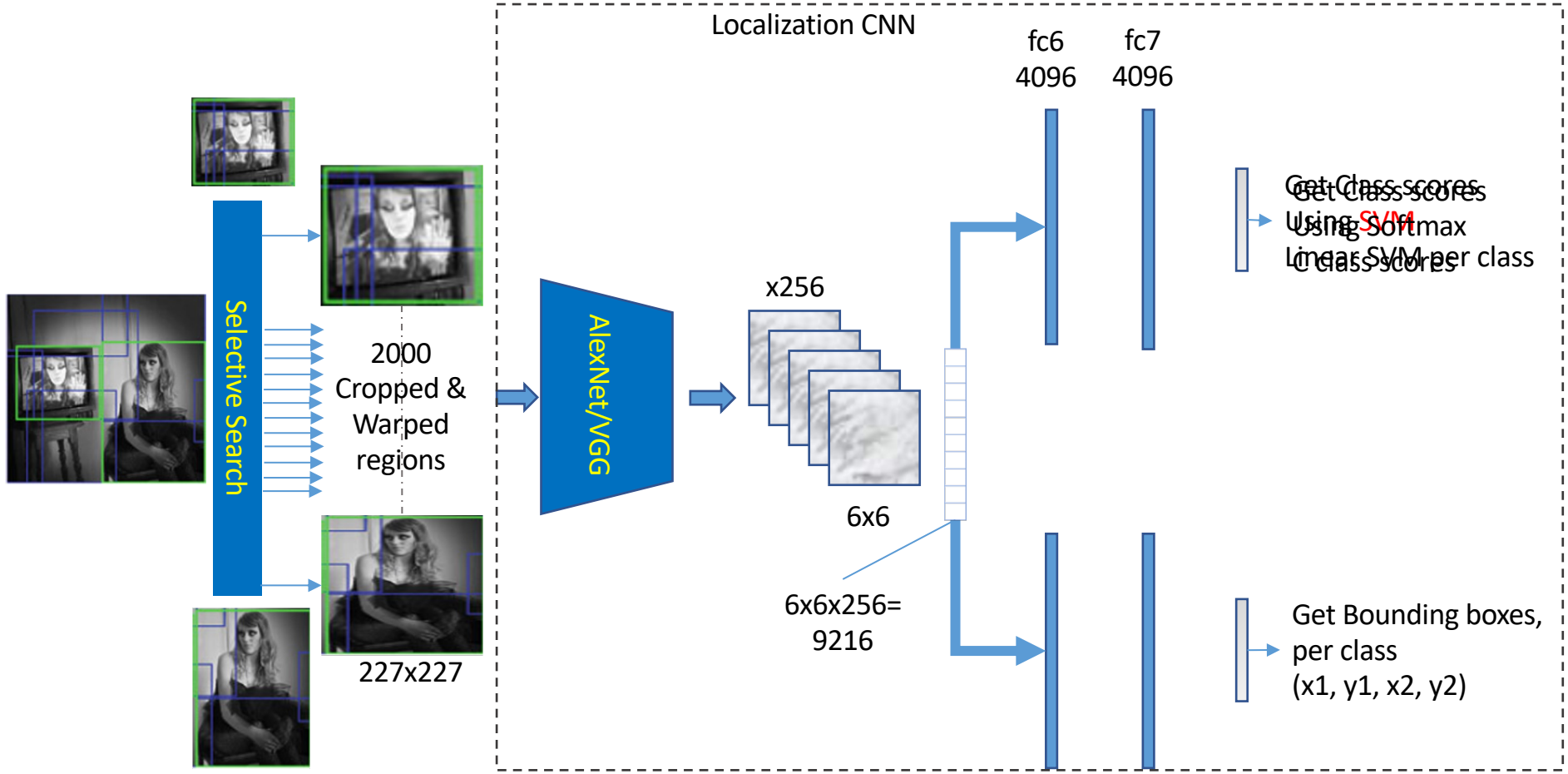
Ross Girshick, et al. from UC Berkeley titled "Rich feature hierarchies for accurate object detection and semantic segmentation."

RCNN - Region proposals with CNNs



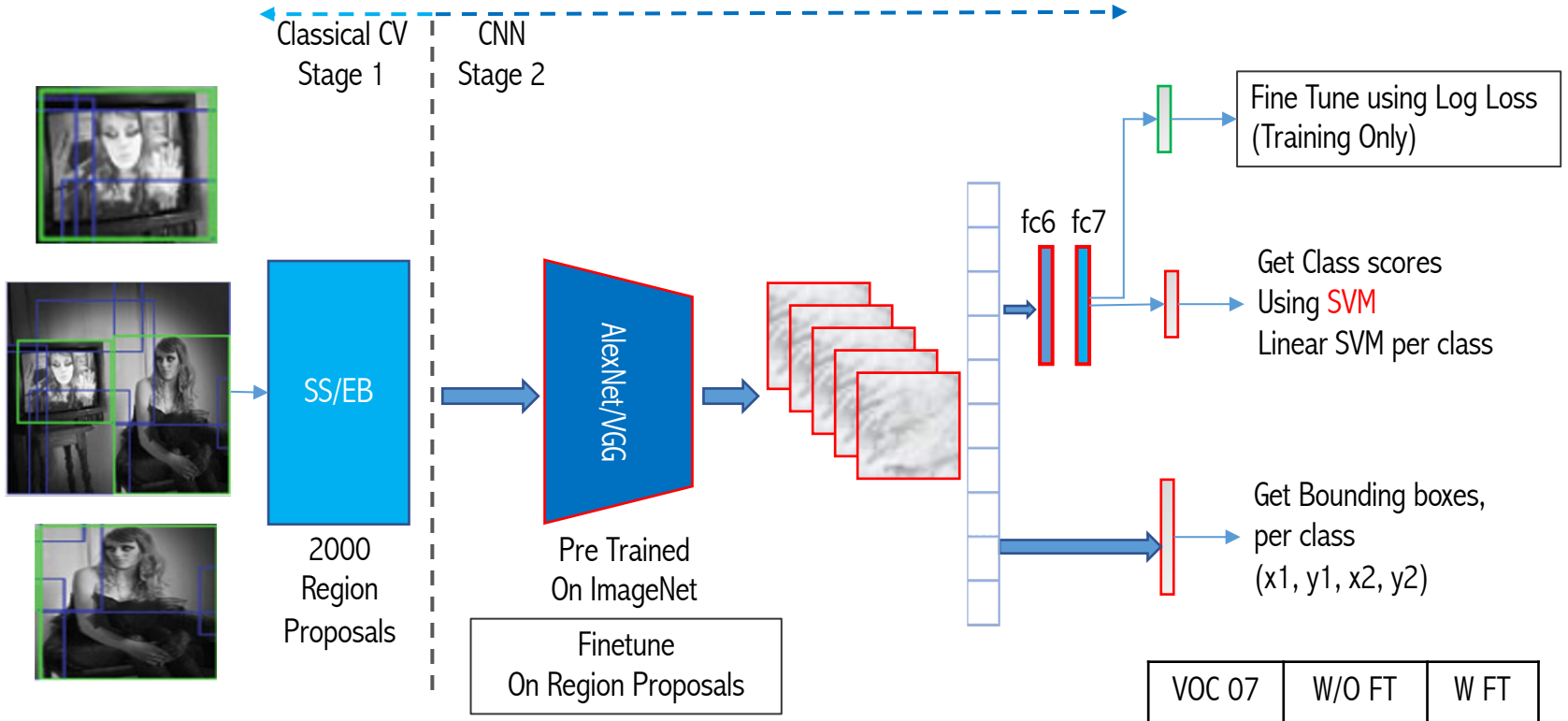
Rich feature hierarchies for accurate object detection and semantic segmentation - Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik

RCNN - Region proposals with CNNs



Rich feature hierarchies for accurate object detection and semantic segmentation - Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik

RCNN - Region proposals with CNNs



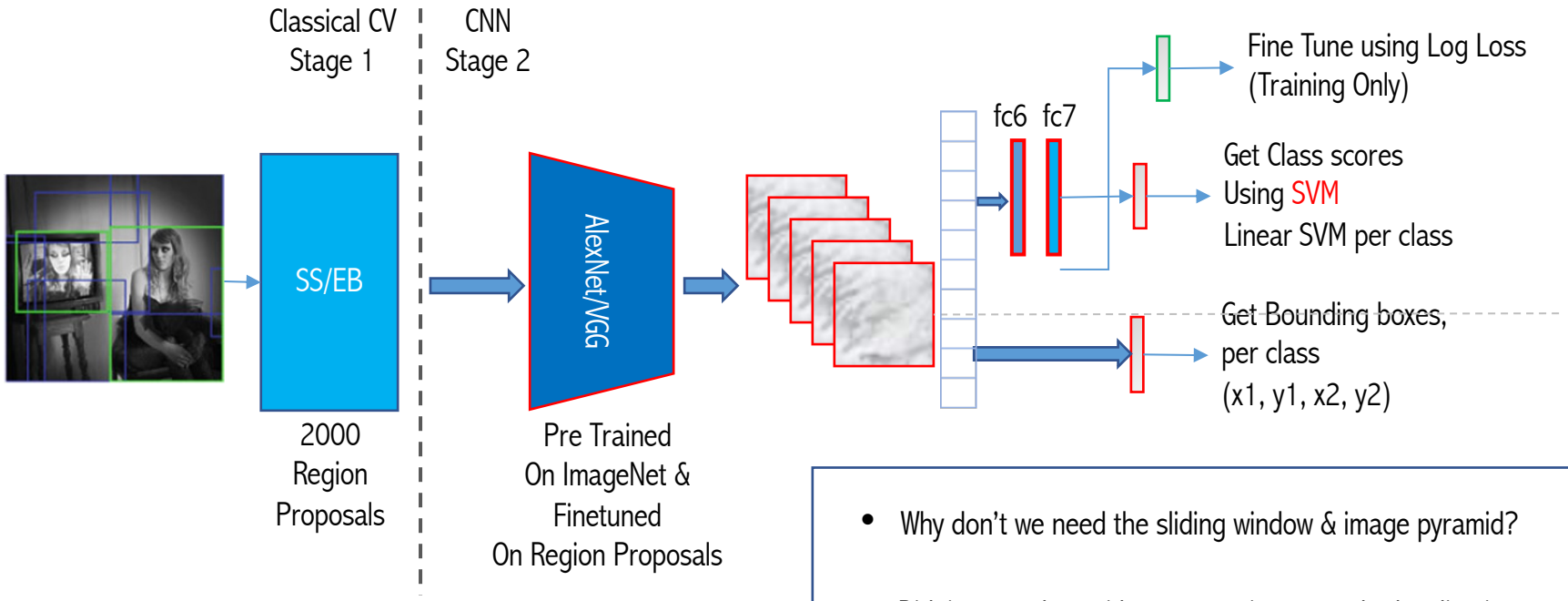
VOC 07	W/O FT	W FT
pool5	44.2	47.3
fc6	46.2	53.1
fc7	44.7	54.2

Alex Net

- Before finetune: 44%
- After finetune: 54%
- Adding bounding box regression: 58%
  - VGG: 66%

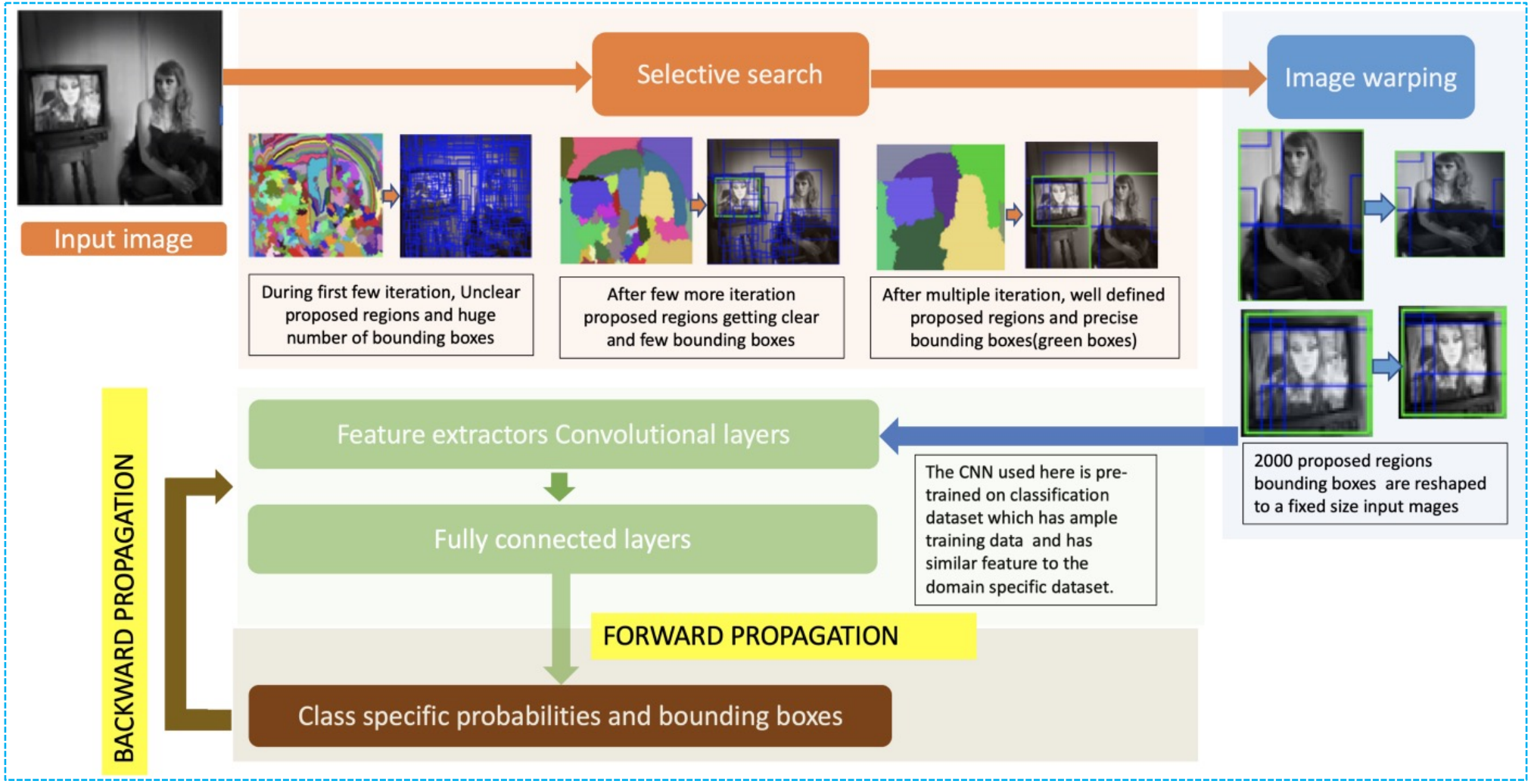
Rich feature hierarchies for accurate object detection and semantic segmentation - Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik

RCNN - Region proposals with CNNs

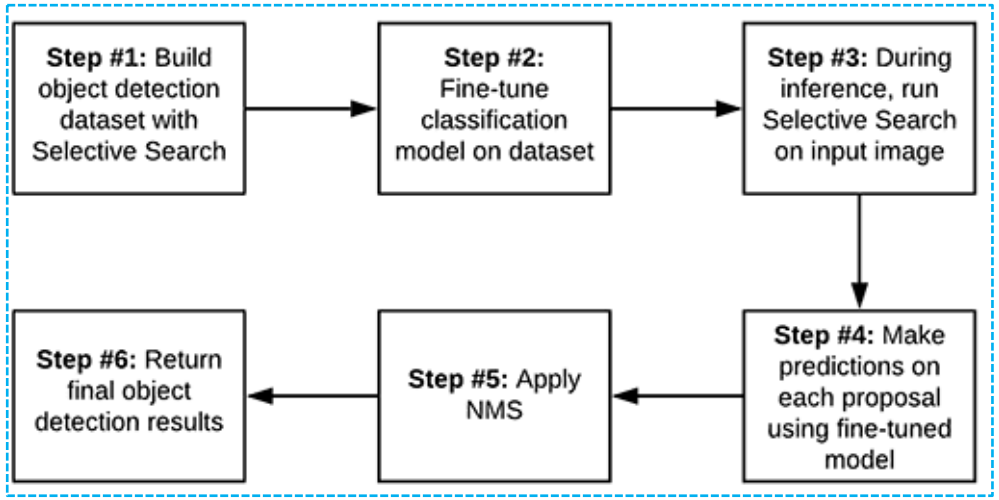


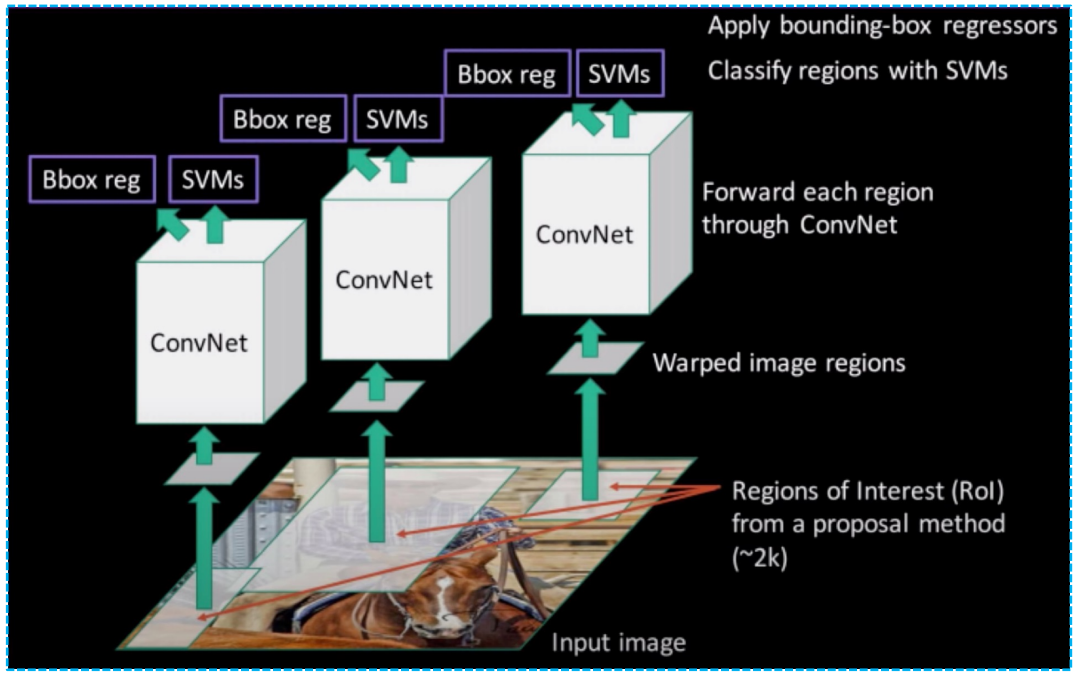
- Why don't we need the sliding window & image pyramid?
- Didn't we end up with too many inputs to the localization network?

RCNN - Region proposals with CNNs



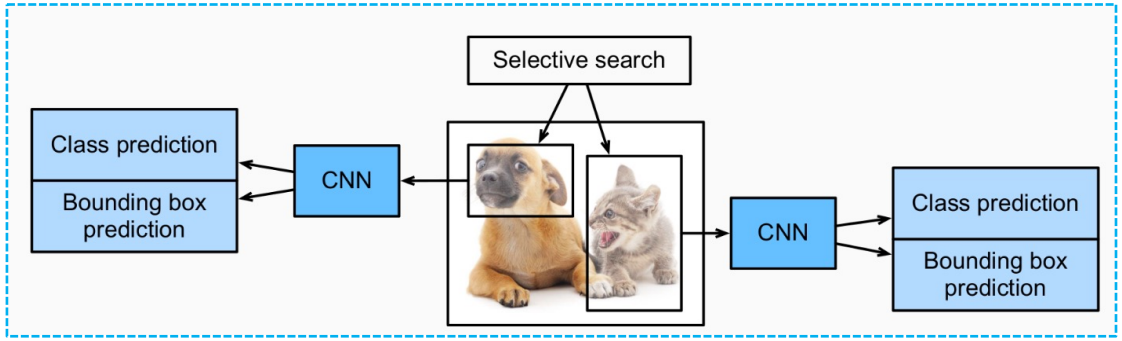
RCNN - Region proposals with CNNs





1.It consumes a huge amount of time, storage, and computation power.
2.It has a complex multi-stage training pipeline(3 stage — Log loss, SVM, and BBox Regressor's L2 loss).

Why not consider running the CNN just once per image and then find a way to share that computation across the ~2000 proposals?

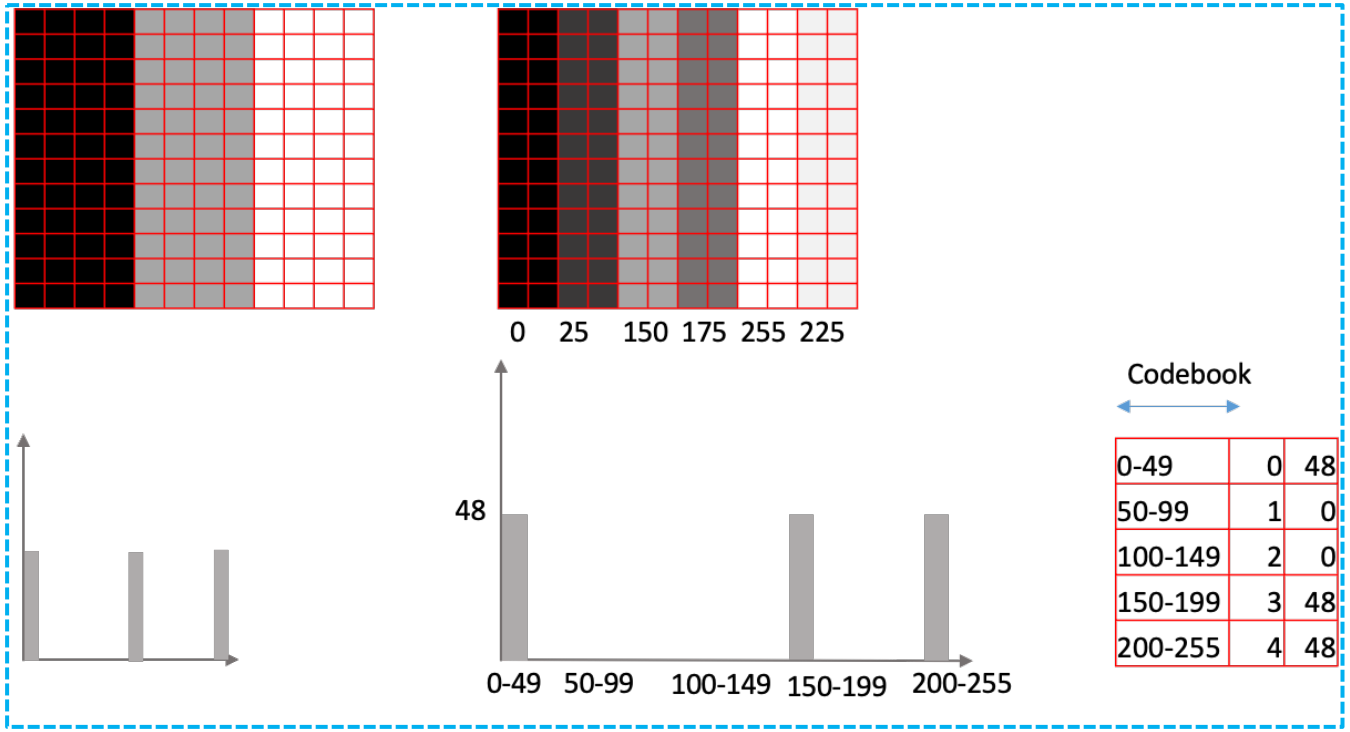


Outline

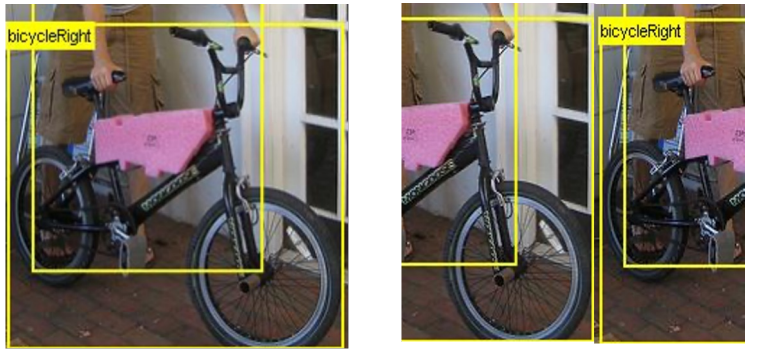
- CNN Limitations
- Region Based Convolutional Neural Networks
- Spatial Pyramid Pooling
- Fast R-CNN
- Faster RCNN
- Object Detection with Transformers: Motivation

Spatial Pyramid Matching

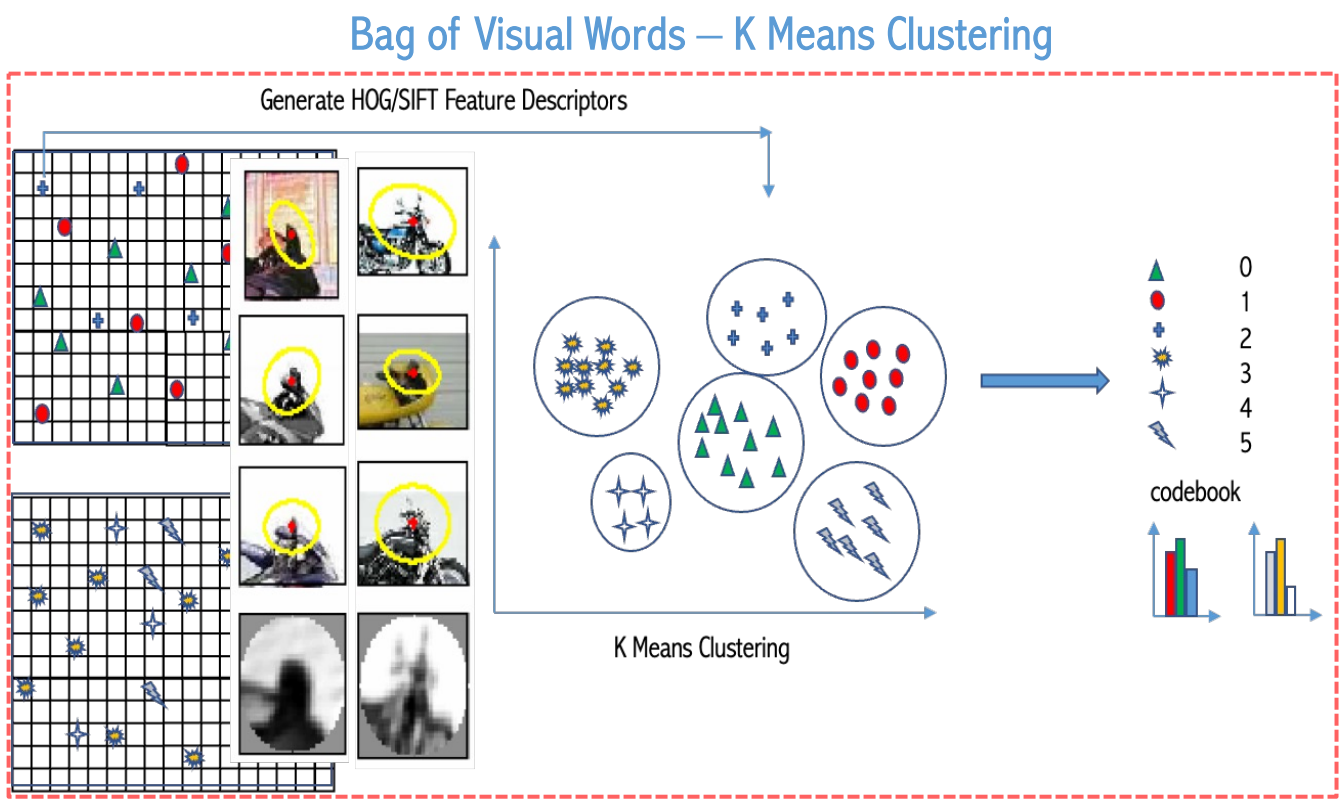
Histograms of Images - Bins

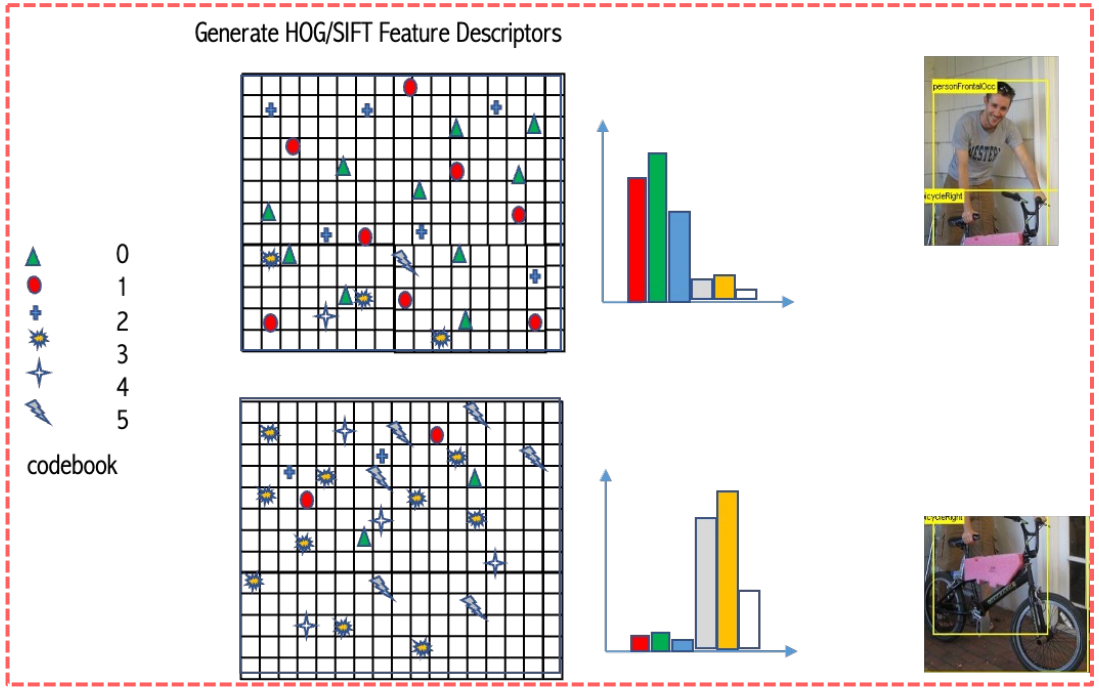


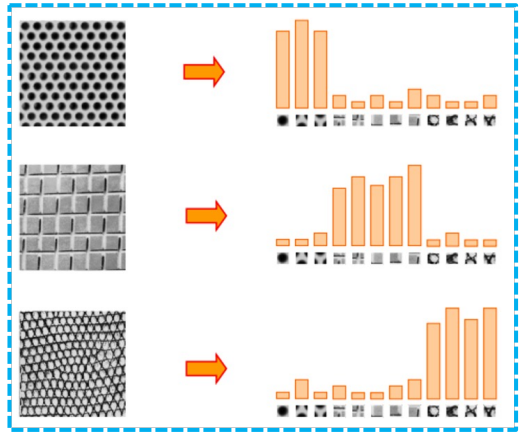
0-49	0	48
50-99	1	0
100-149	2	0
150-199	3	48
200-255	4	48



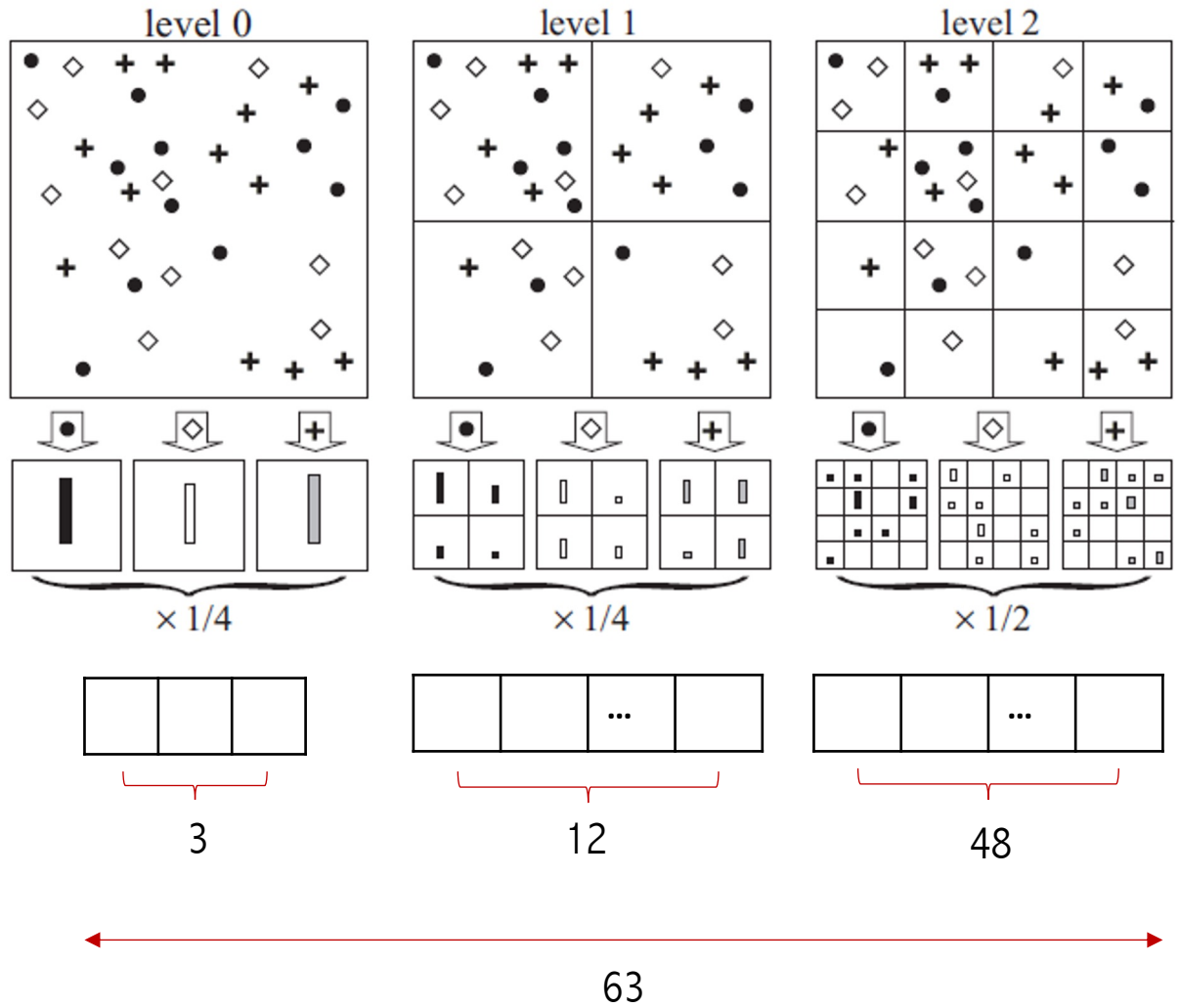
Spatial Pyramid Matching



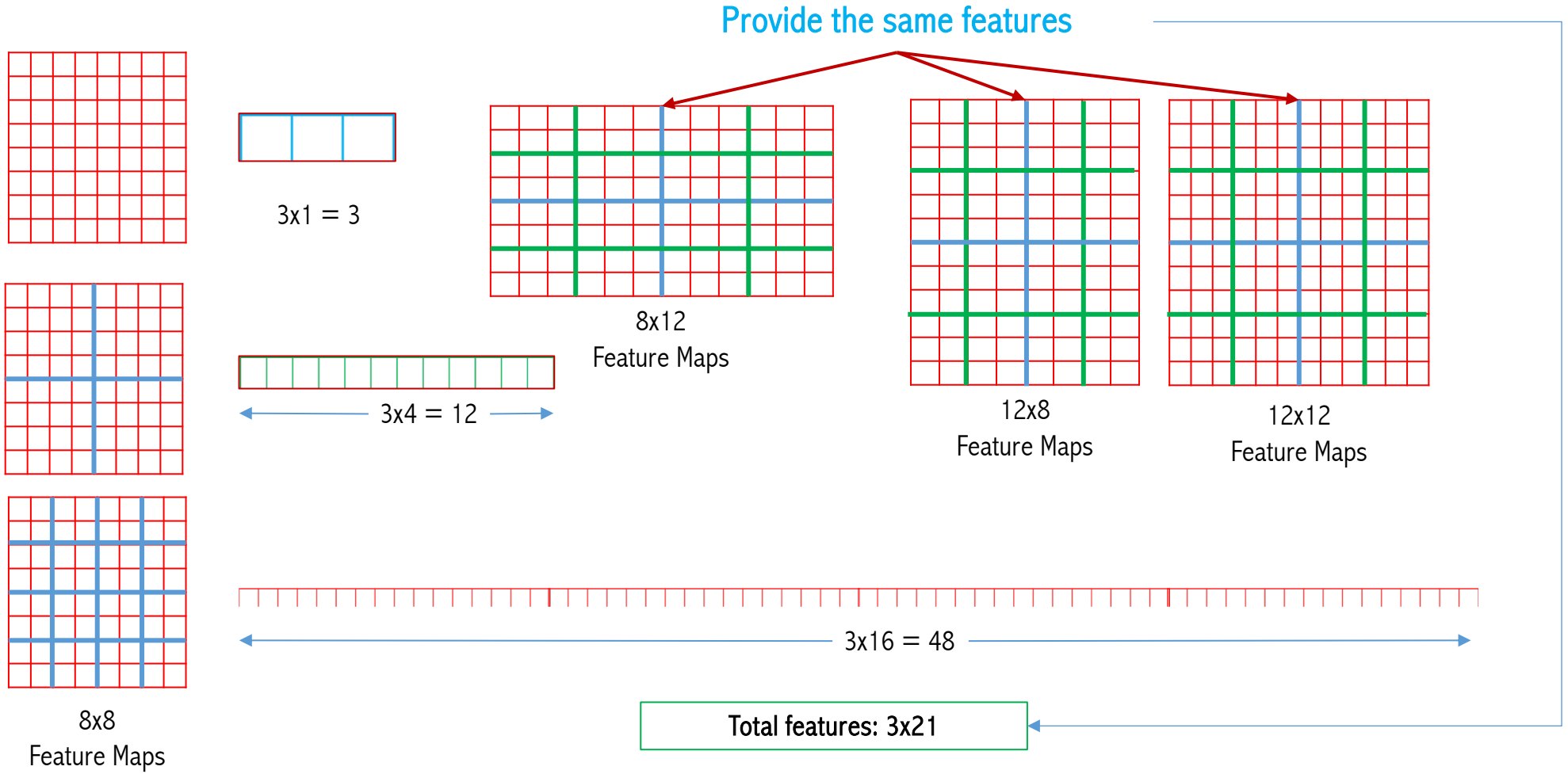




Spatial Pyramid Matching

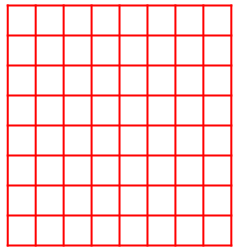


Spatial Pyramid Matching



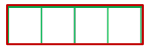
Any Size and Aspect Ratio

Spatial Pyramid Pooling



8x8
Feature Maps

1x1



4x1

16x1

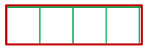
21x1

- Identifying features
- K-means clustering
- Codebooks
- Histograms

Just Max-Pool

x256

1x256

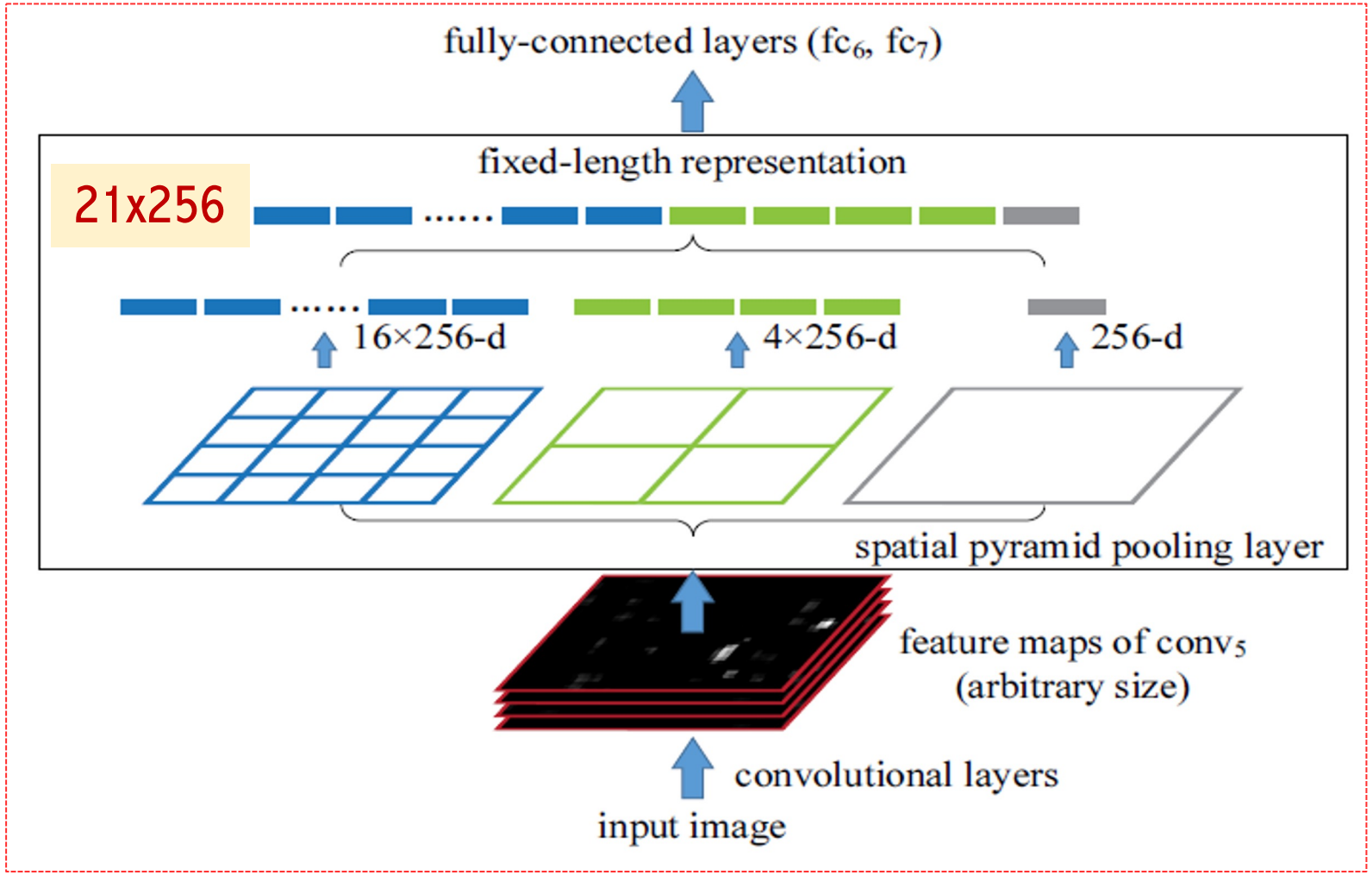


4x256

16x256

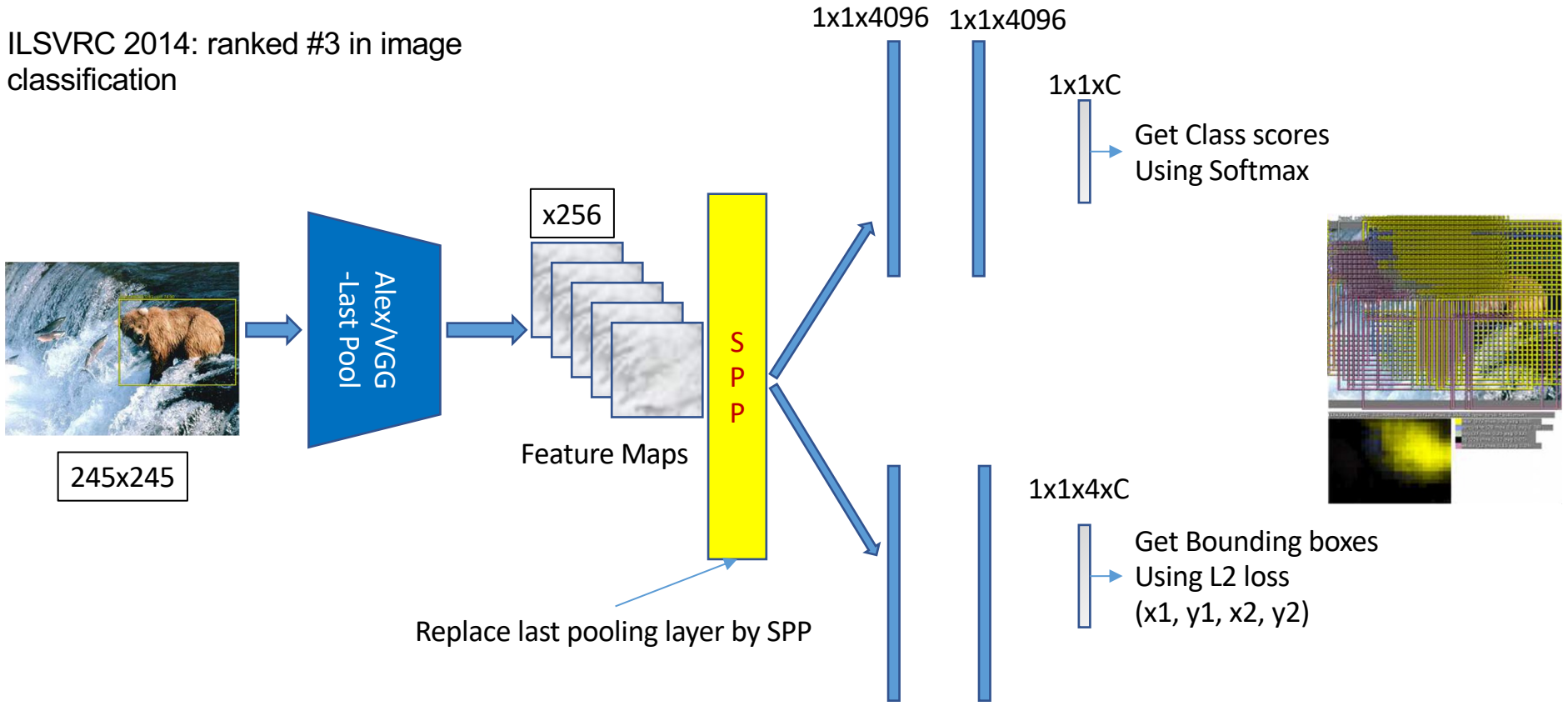
21x256

Spatial Pyramid Pooling

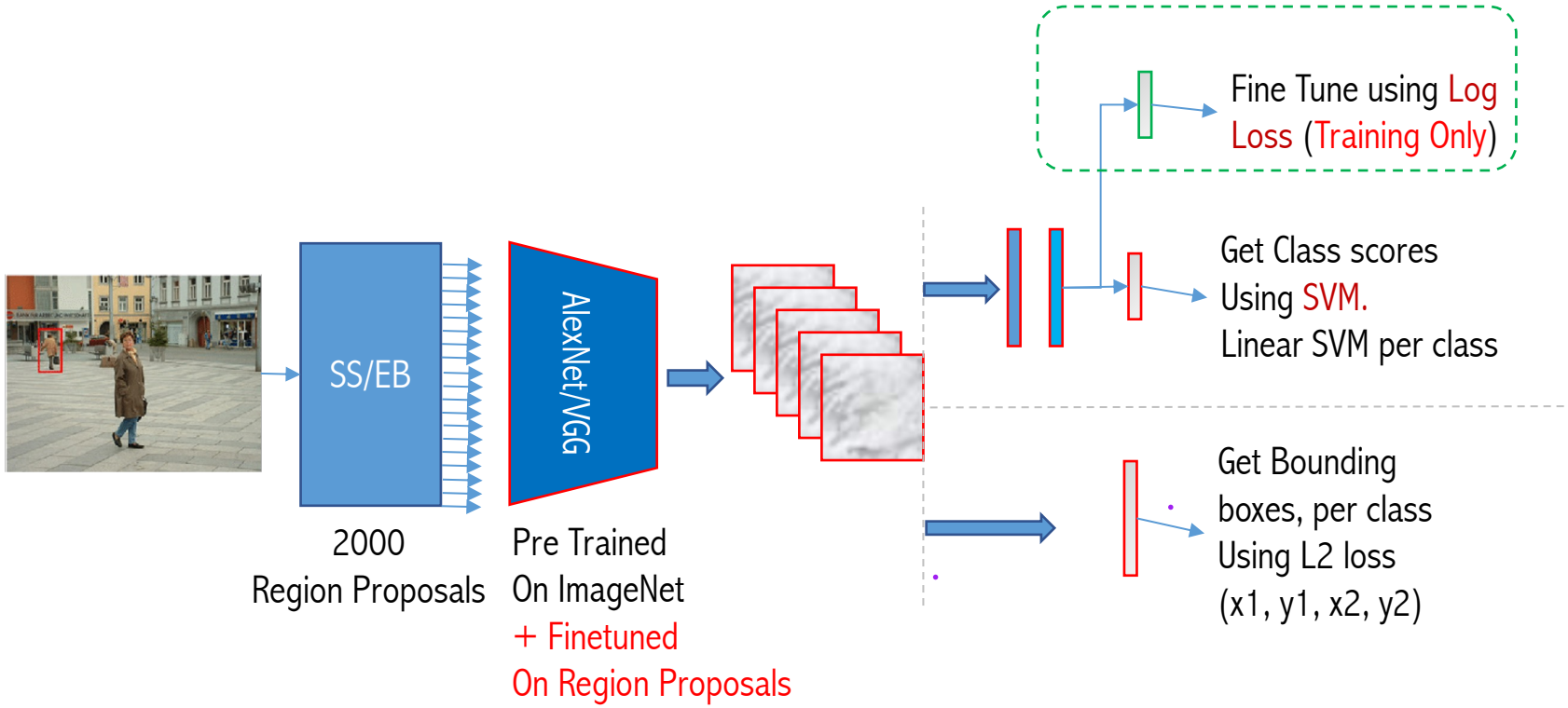


SPPNet = SPP + Overfeat for Classification

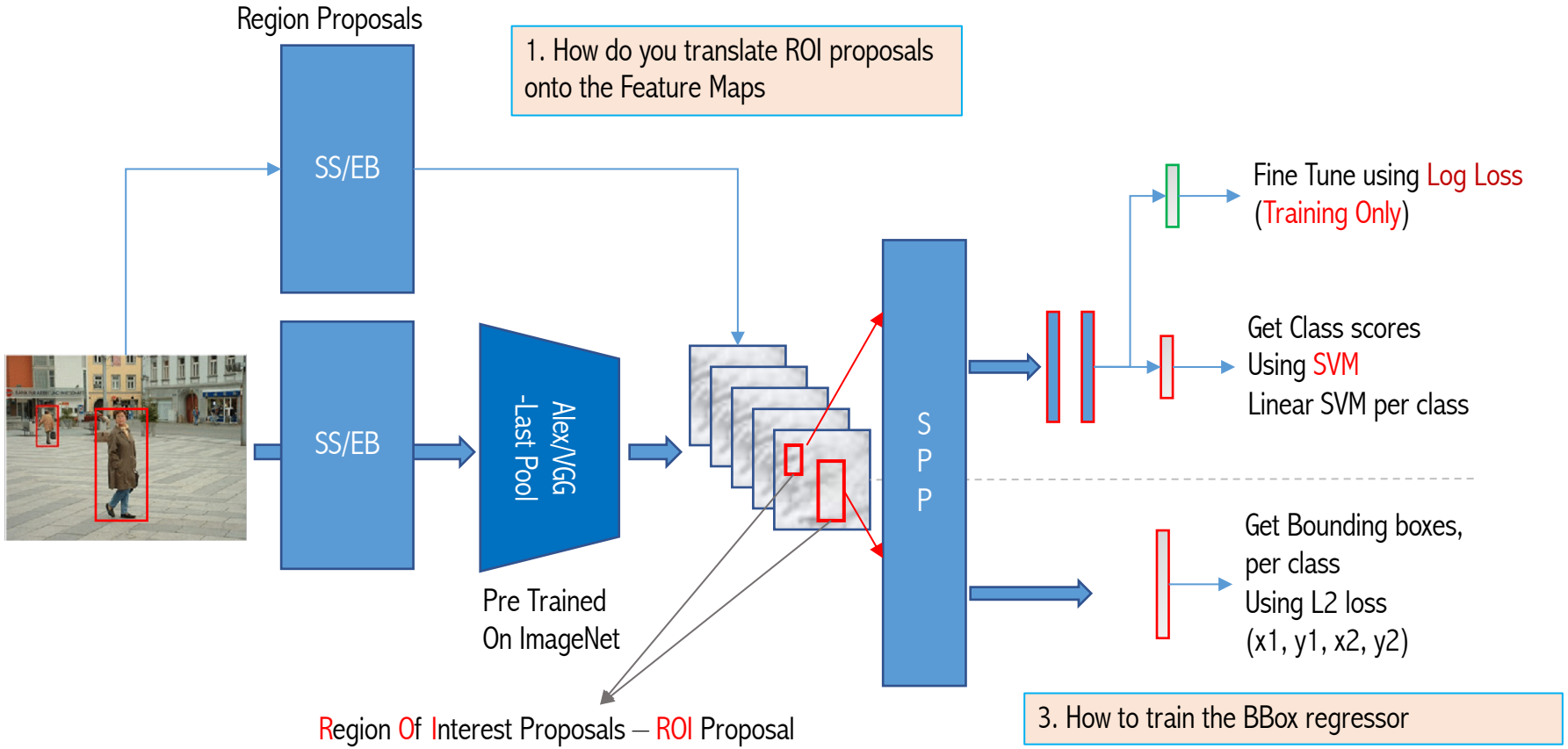
ILSVRC 2014: ranked #3 in image classification



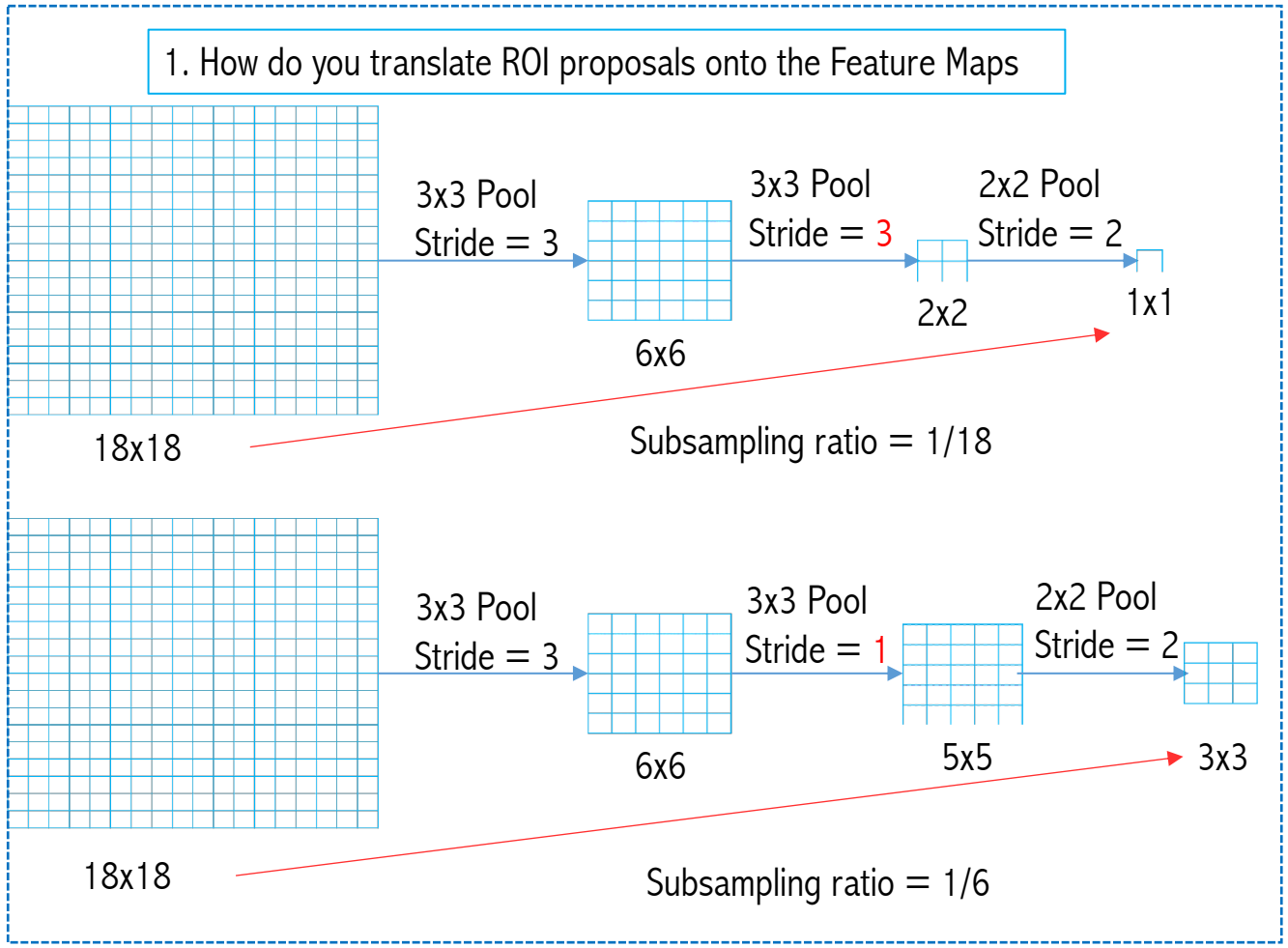
RCNN – Two stage-based methods

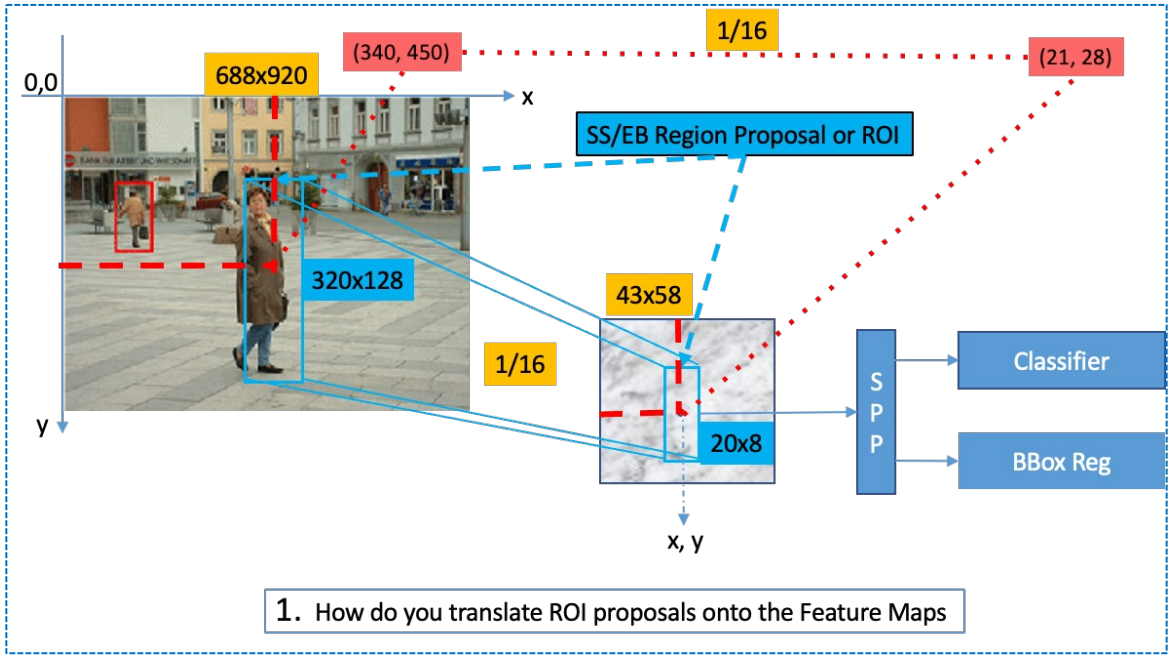


SPP – Two stage-based methods

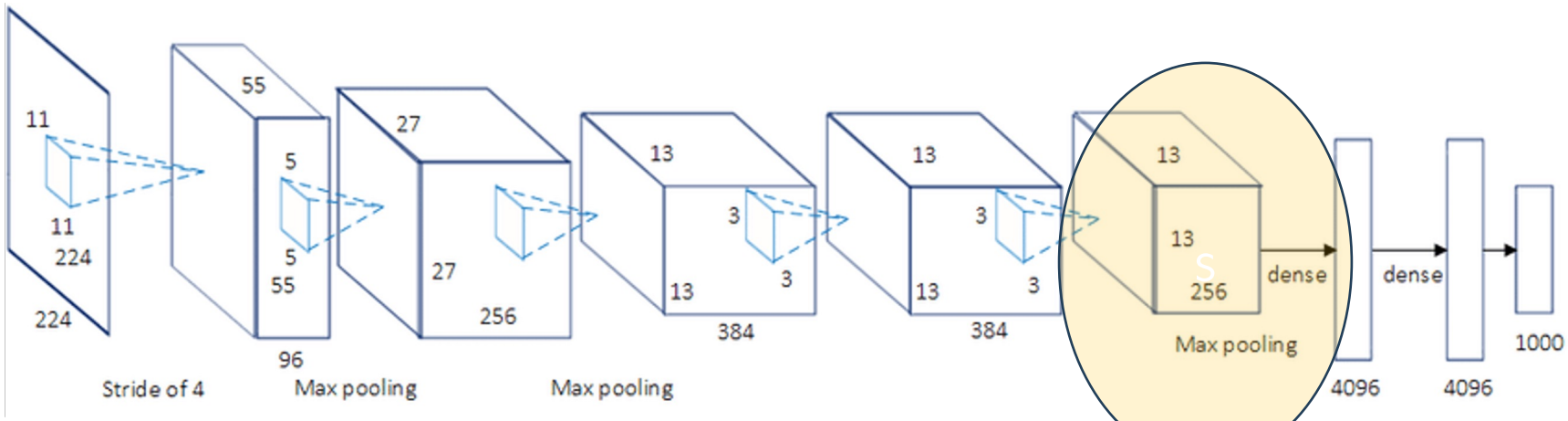


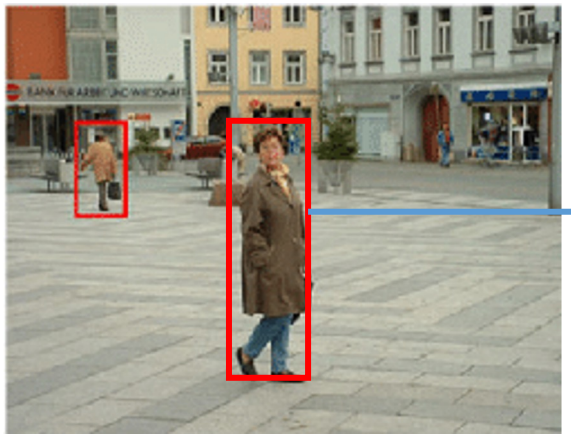
Subsampling Ratio

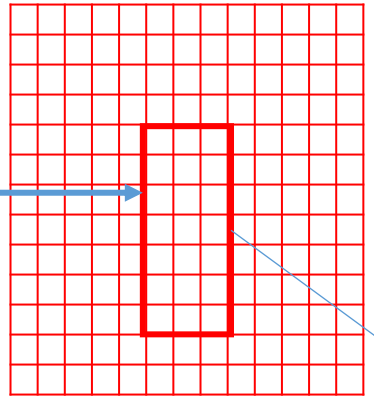




AlexNet Subsampling Ratio







2. How do you pool the ROI proposals from the Feature Map

13x13
Feature Maps

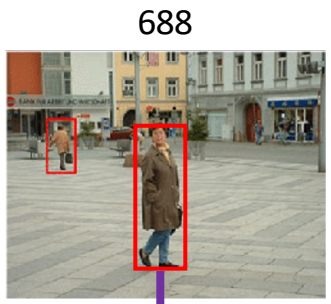
Spatial Pyramid Pooling
Level 1, Level 2, Level 3

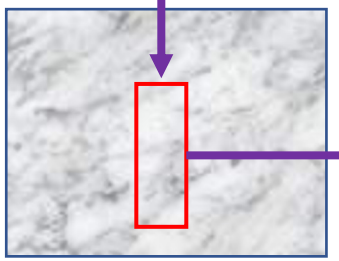
6x6 grid. It is impossible

How to solve

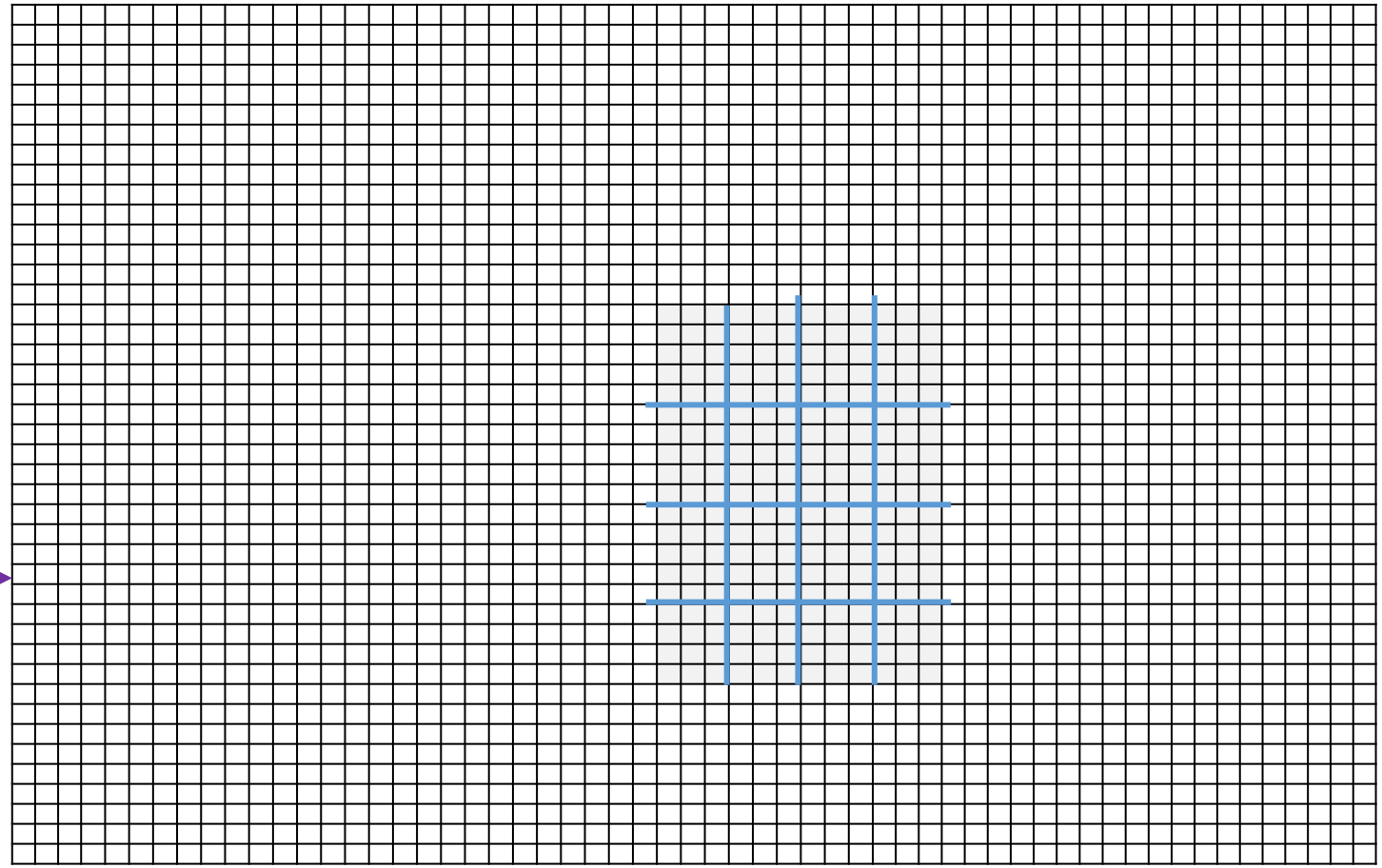
SPP on Region Proposals

Aspect ratio 3:4



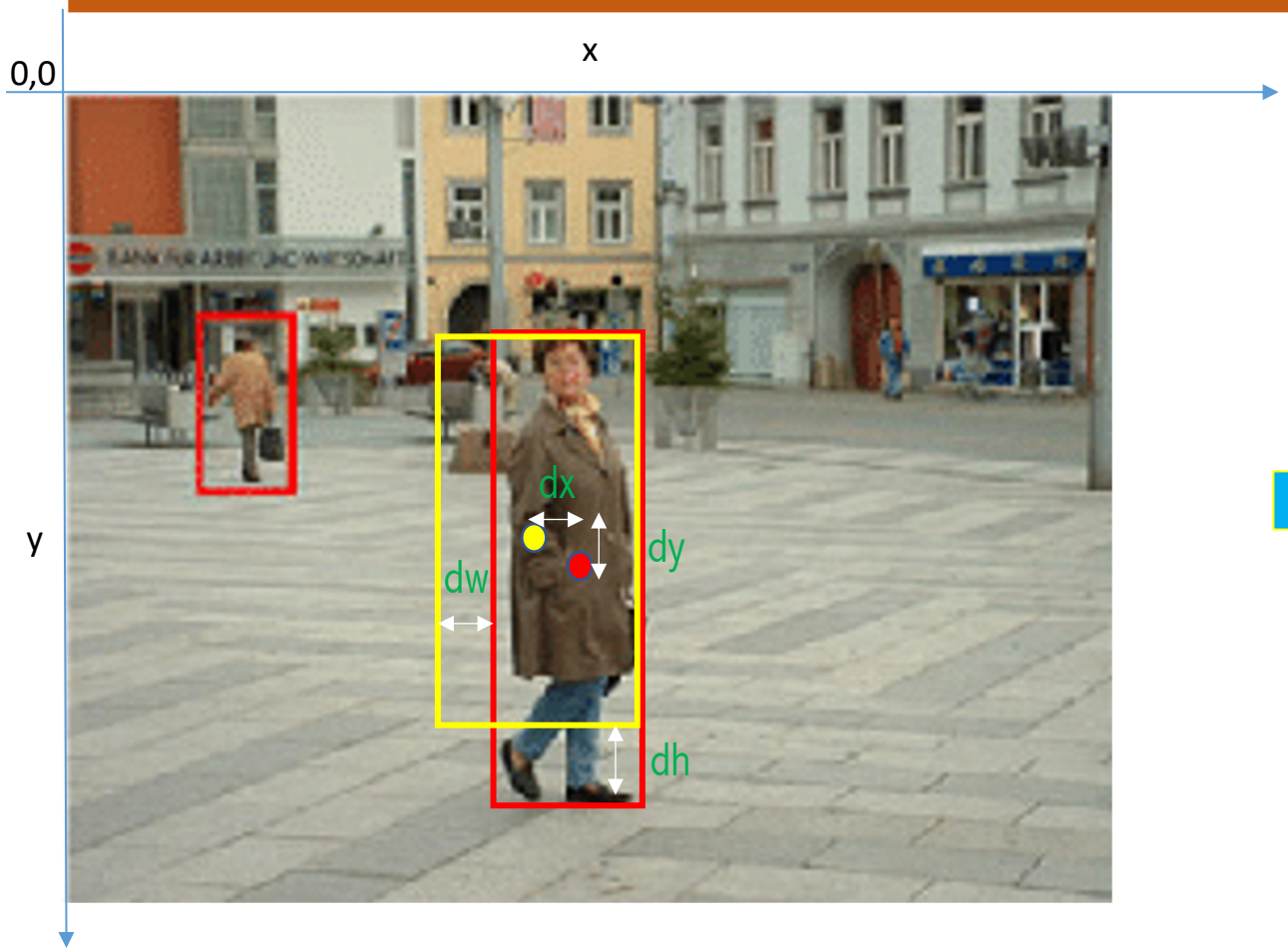


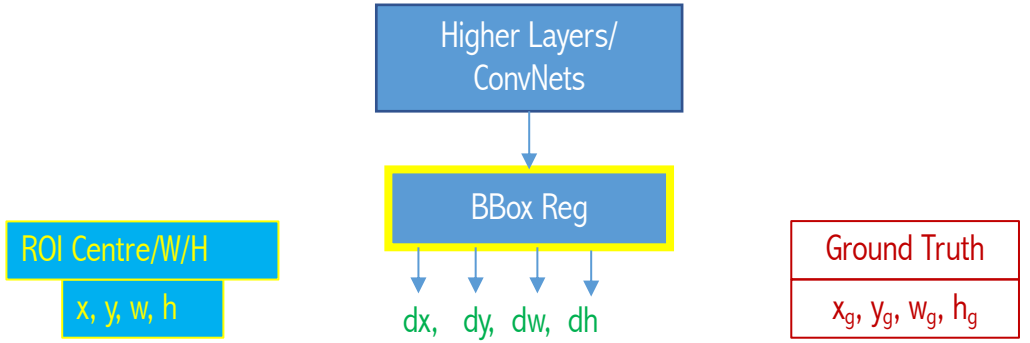
43x58 - Feature Map



Three level in Practice - {6x6, 3x3, 2x2, 1x1}

BBox Regression Training

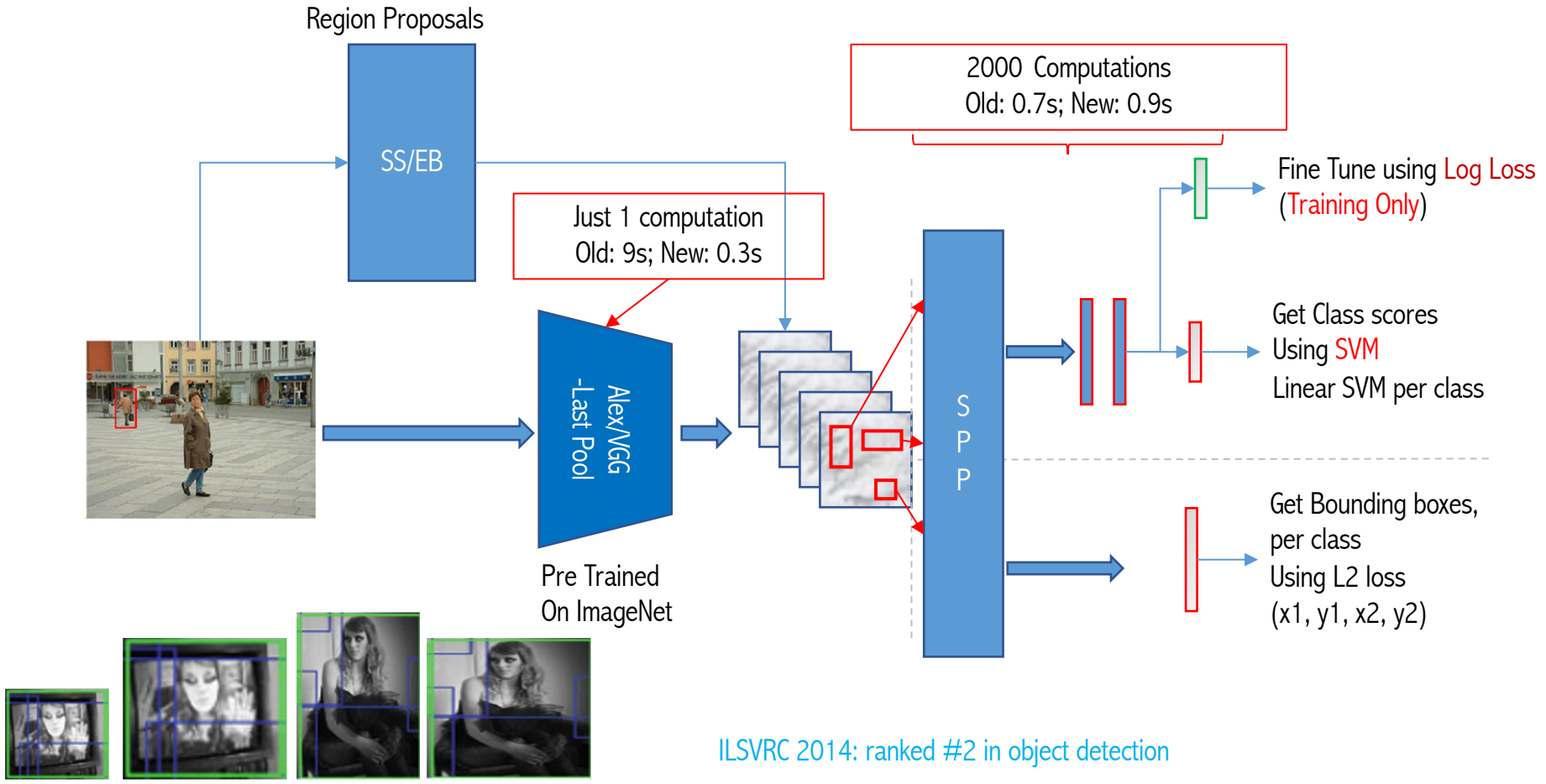




$$(x + dx - x_g)^2 = 0$$

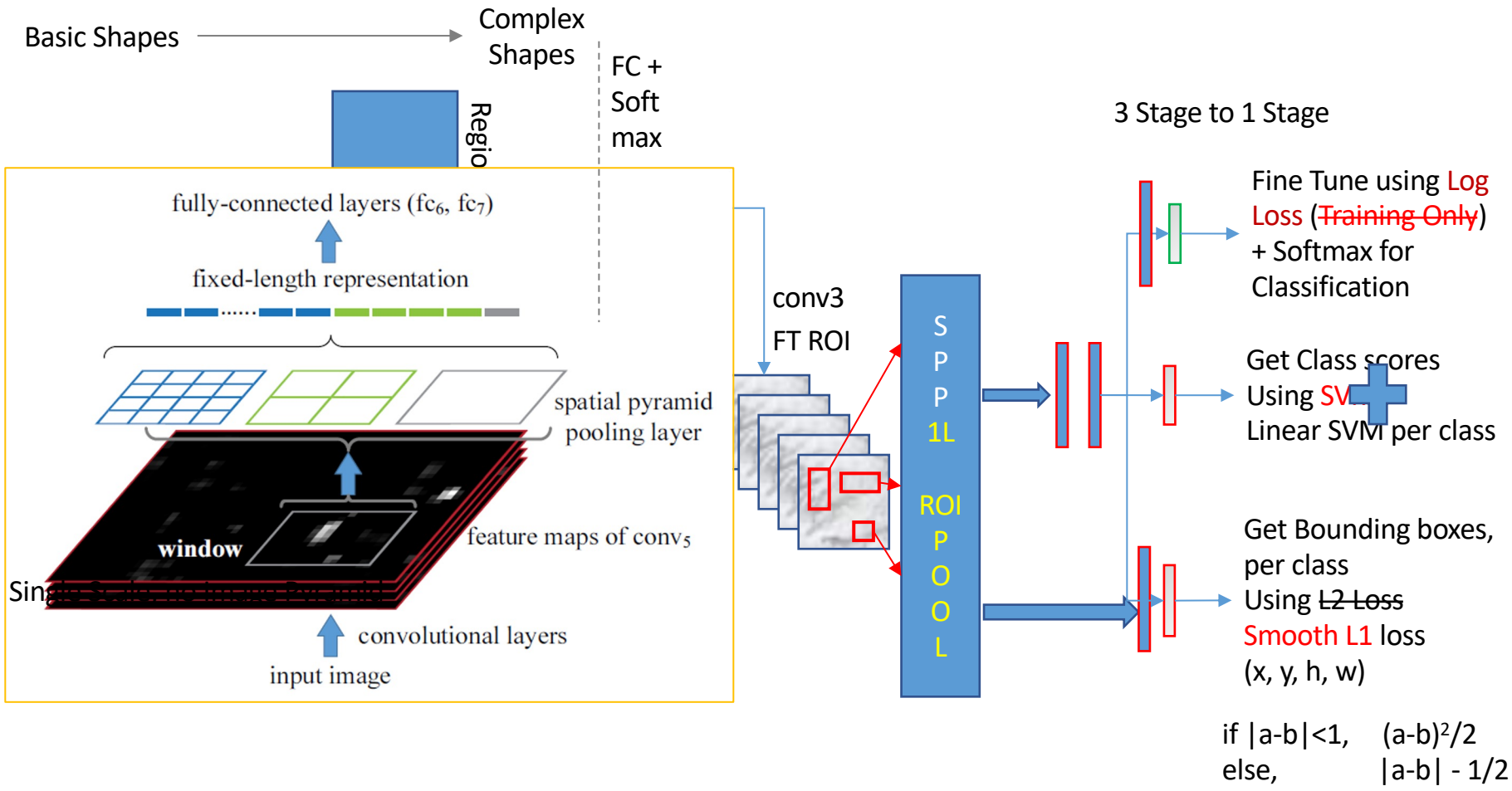
3. How to train the BBox regressor

SPP – 2 Stage Network - Inference



ILSVRC 2014: ranked #2 in object detection

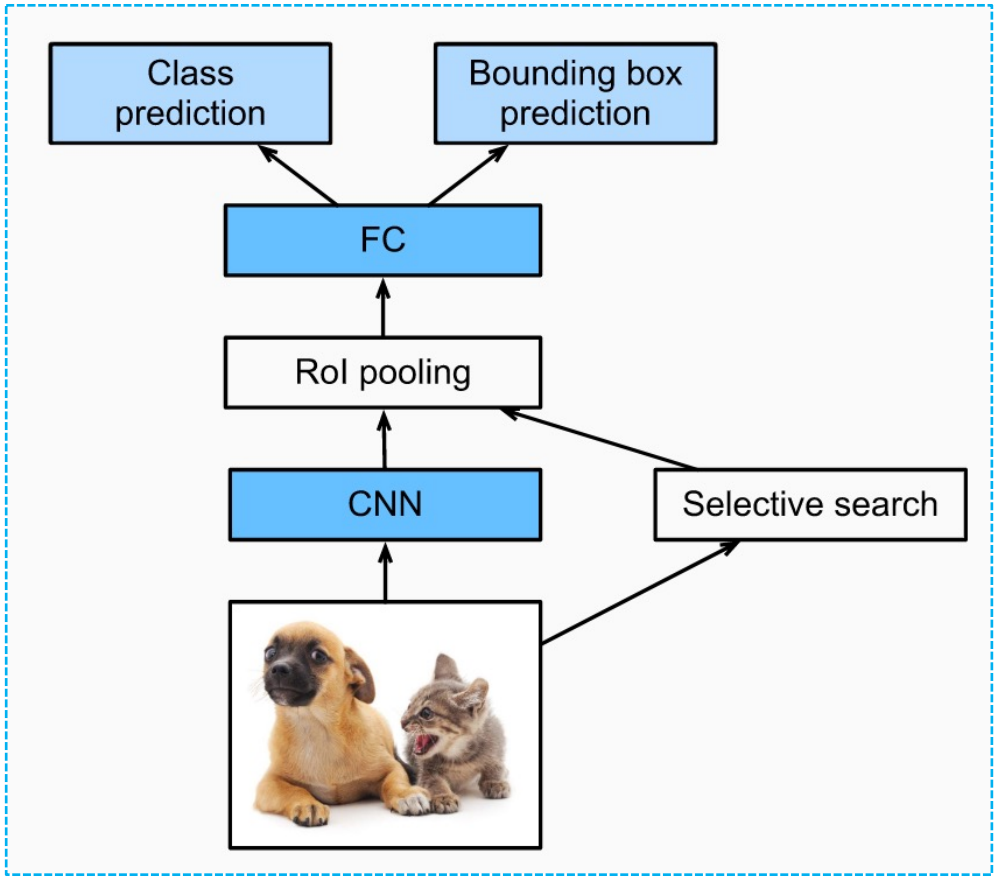
Speed-up: SPP vs RCNN=>Fast RCNN

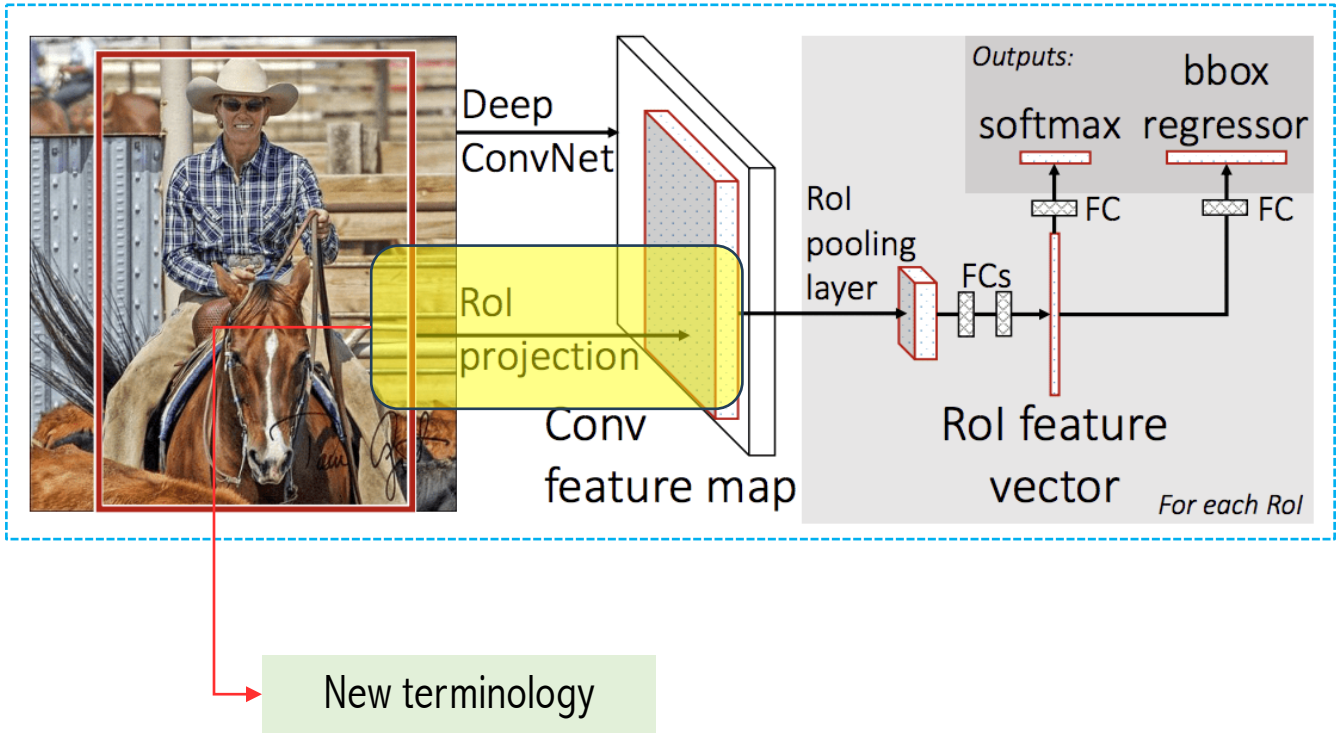


	SPP (1-sc) (ZF-5)	SPP (5-sc) (ZF-5)	R-CNN (Alex-5)
$pool_5$	43.0	44.9	44.2
fc_6	42.5	44.8	46.2
$ftfc_6$	52.3	53.7	53.1
$ftfc_7$	54.5	55.2	54.2
$ftfc_7$ bb	58.0	**59.2**	58.5
conv time (GPU)	0.053s	0.293s	8.96s
fc time (GPU)	0.089s	0.089s	0.07s
total time (GPU)	0.142s	0.382s	9.03s
speedup (*vs.* RCNN)	**64**×	**24**×	-

Table 9: Detection results (mAP) on Pascal VOC 2007. "ft" and "bb" denote fine-tuning and bounding box regression.

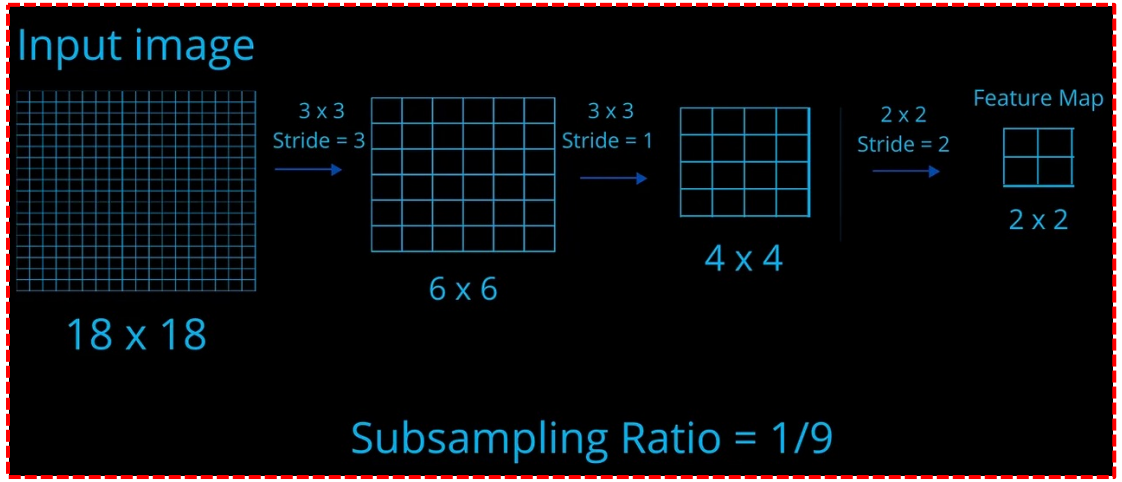
Fast R-CNN

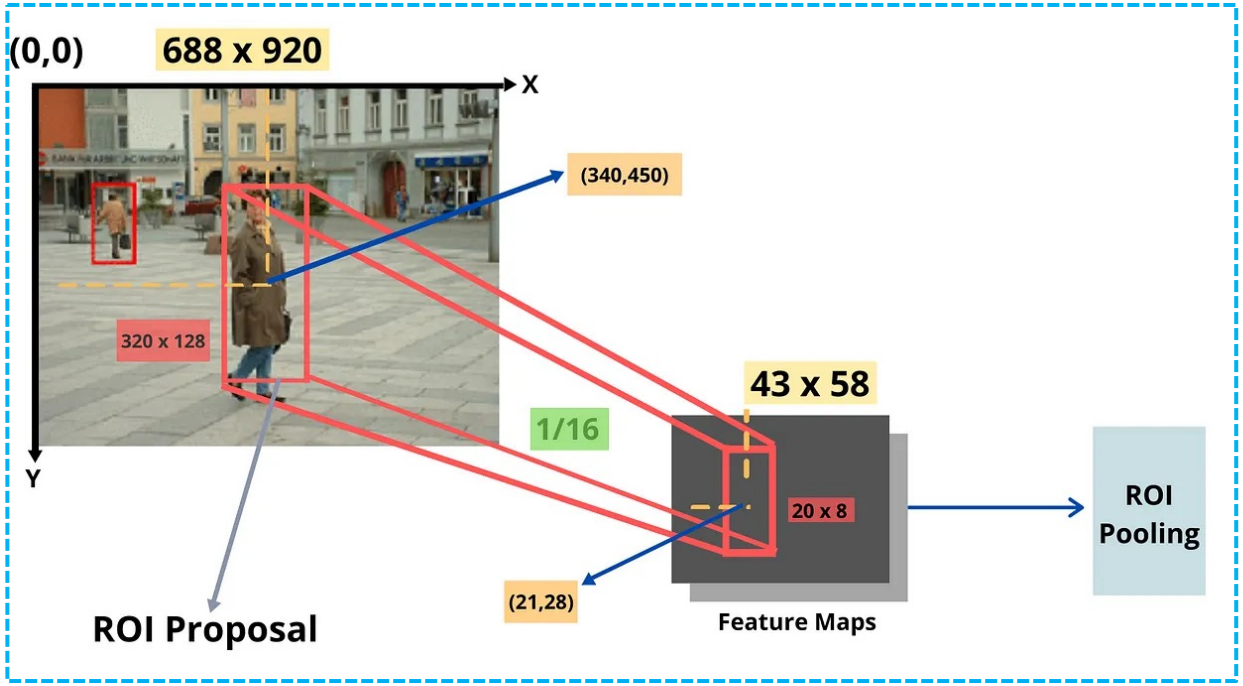




Fast R-CNN, which was developed a year later after R-CNN, solves these issues very efficiently and is about 146 times faster than the R-CNN during the test time.

Sub-Sampling Ratio & Roi Projection



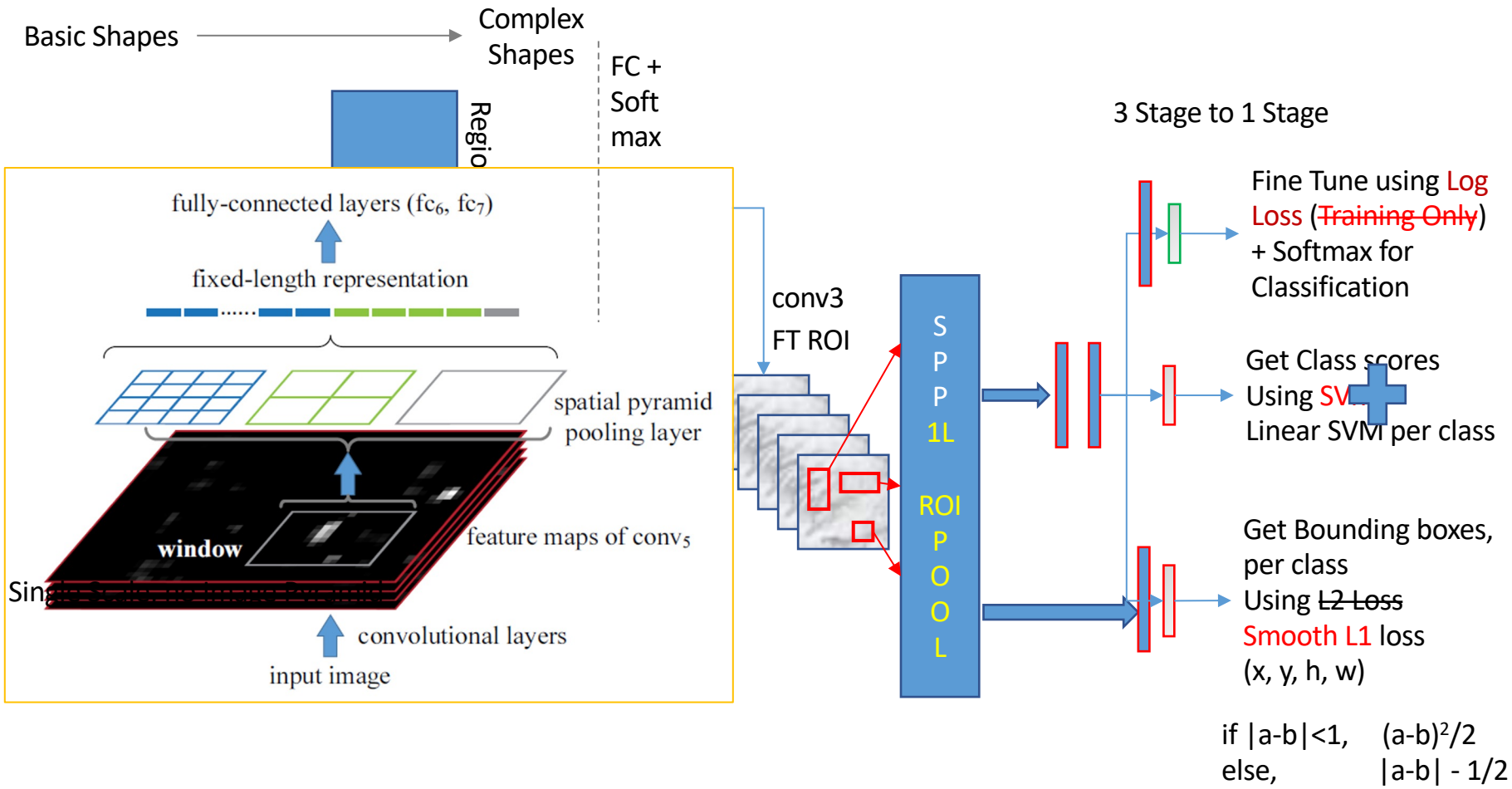


The idea of ROI projection is that we get the coordinates of the bounding box from the ROI proposal and we need to project them onto the feature maps by projecting the ROI proposal with respect to the subsampling ratio.

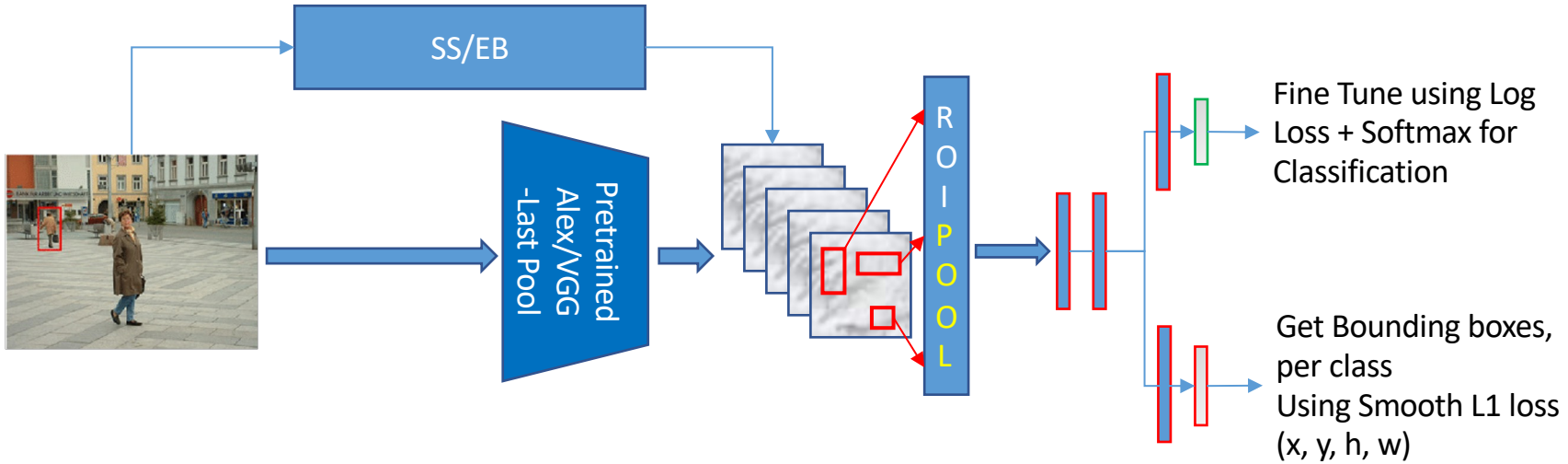
Outline

- CNN Limitations
- Region Based Convolutional Neural Networks
- Spatial Pyramid Pooling
- Fast R-CNN
- Faster RCNN
- Object Detection with Transformers: Motivation

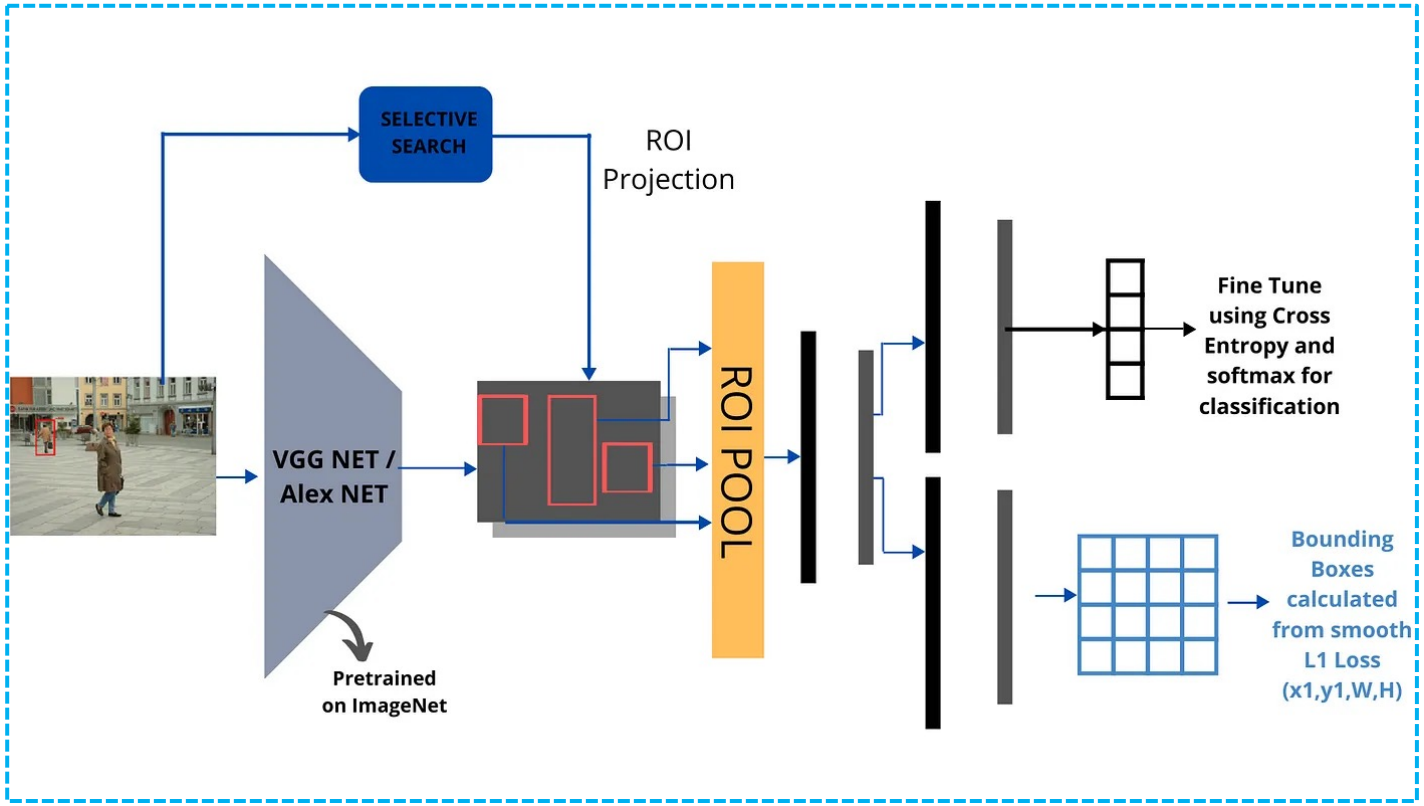
RCNN -> SPPNet -> Fast RCNN

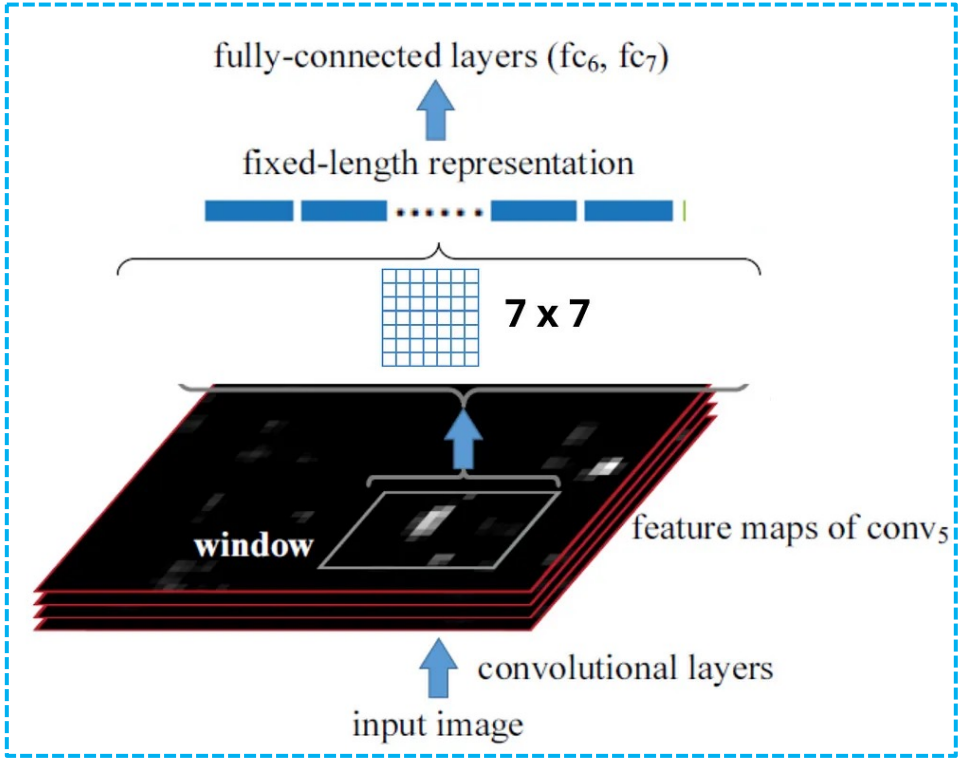


RCNN -> SPPNet -> Fast RCNN



Fast R-CNN





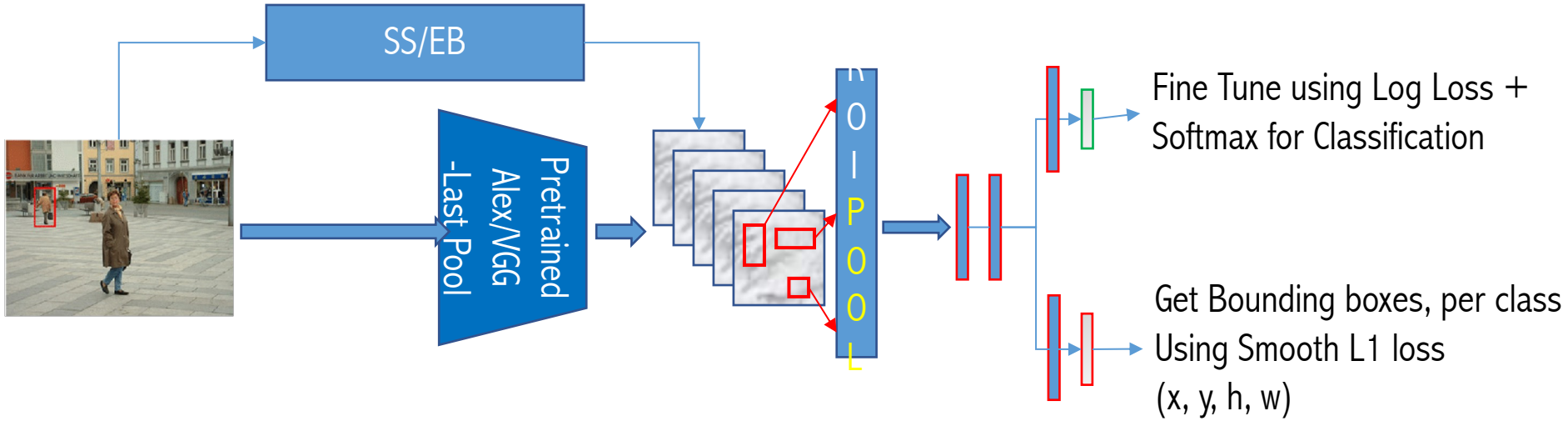
	RCNN	Fast RCNN
Training Time	84 hours	9.5 hours
(Speedup)	1x	8.8x
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x
Test time per image with Selective Search	50 seconds	2 seconds
(Speedup)	1x	25x

So why don't we reuse those same CNN results for region proposals instead of running a separate selective search algorithm?

Outline

- CNN Limitations
- Region Based Convolutional Neural Networks
- Spatial Pyramid Pooling
- Fast R-CNN
- Faster RCNN
- Object Detection with Transformers: Motivation

Criteria for replacing SS



< 2000 Region Proposals

As fast as SS or better

As Accurate as SS or better

Should be able to propose Overlapping ROIs with different Aspect Ratios and Scale

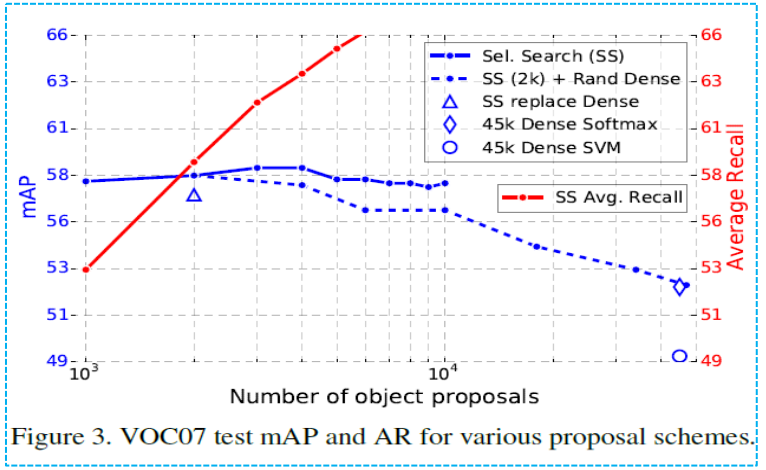
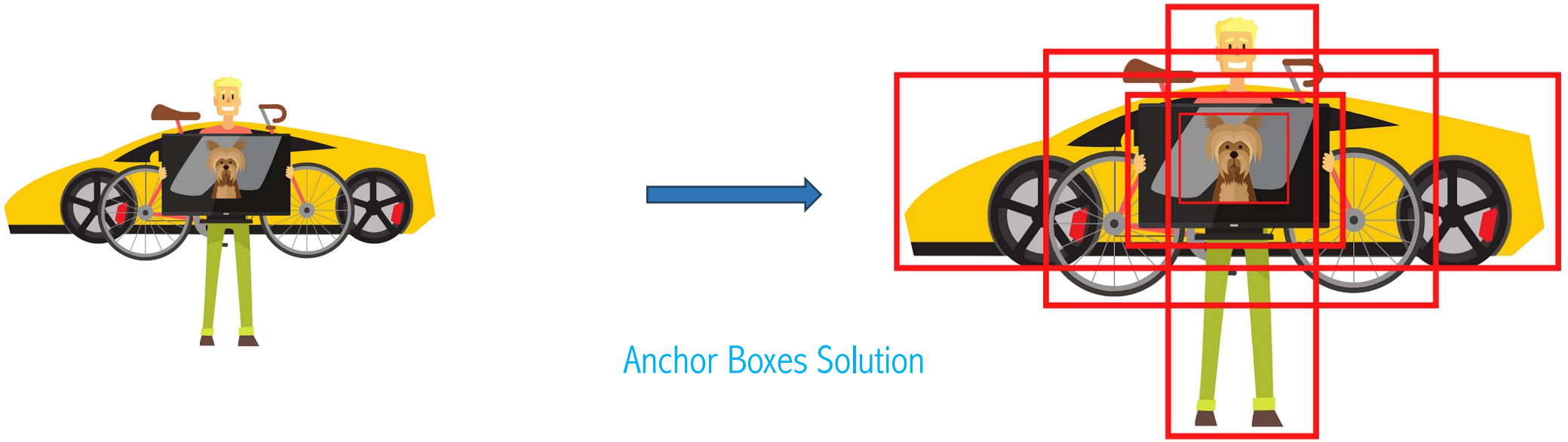


Figure 3. VOC07 test mAP and AR for various proposal schemes.

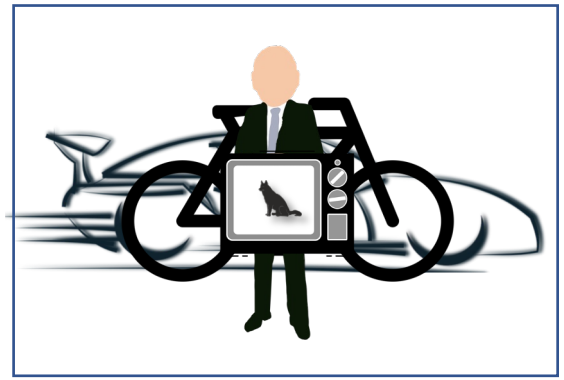
Role of Region Proposals

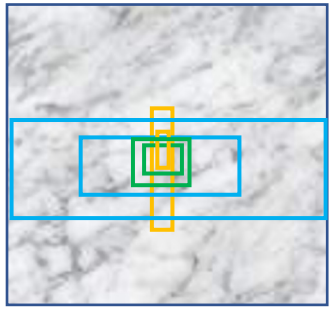
Overlapping ROIs

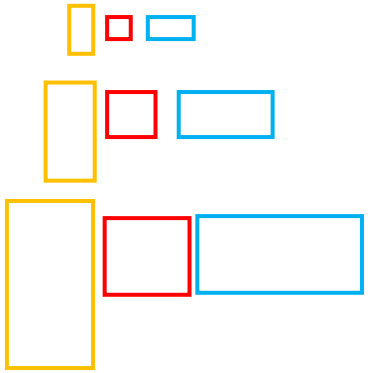


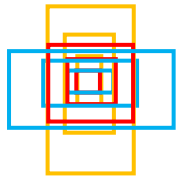
From the image, we see a lot of objects overlapping each other. We see a car, a bicycle, a person holding a television, and a dog inside this television. The selective search could solve this problem but we end up with a huge number of ROIs. We need to think of an idea that efficiently solves this.

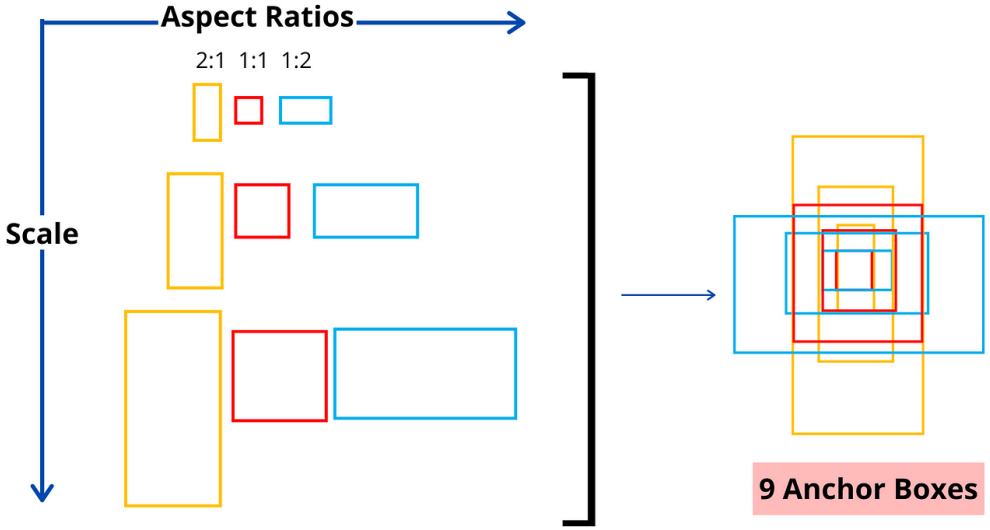
Anchor Boxes









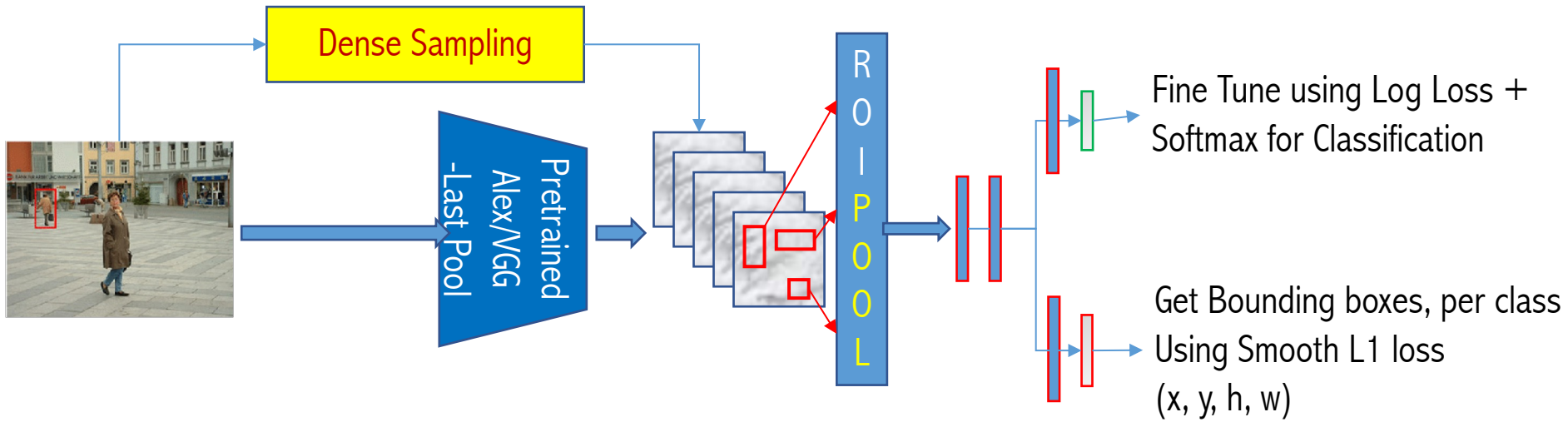


Any object in the image can be detected using boxes of 3 different scales and 3 different aspect ratios.

This could be a technique that can be used to solve our purpose of replacing the region proposer.

Solutions for Replace Region Propsals

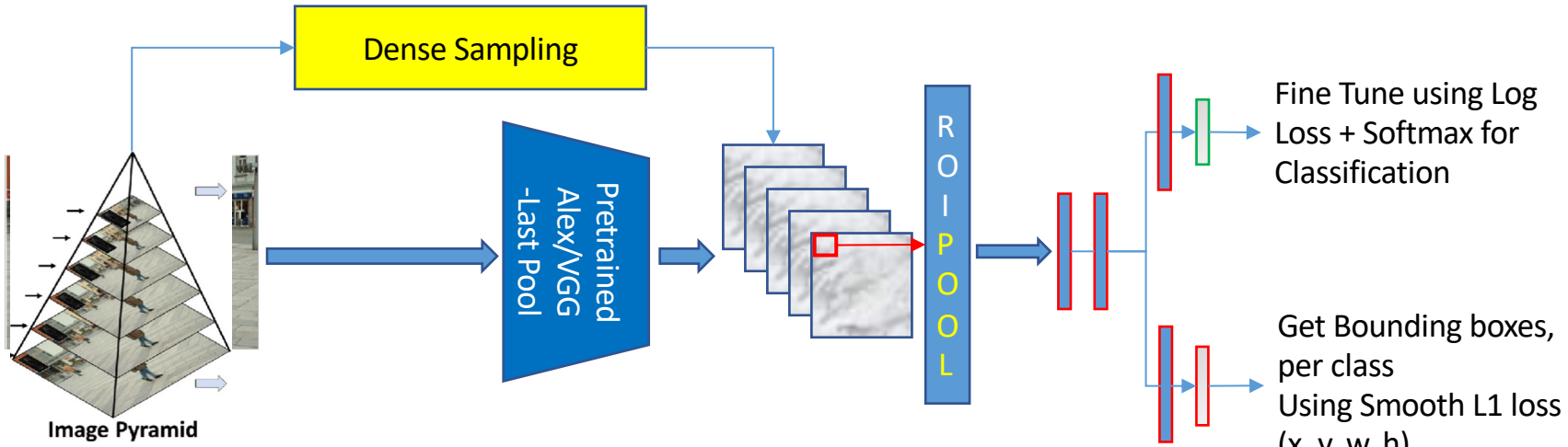
Removing Selective Search and applying a sliding window on top of the Feature Maps. But with this, we end detecting mostly objects of a single scale.



Fast RCNN + Sliding Window

Solutions for Replace Region Propsals

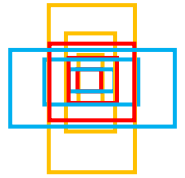
To take care of multiple scales, we have to use Image Pyramids at the input. But using images of 5 different scales (by which almost every object can be detected) makes the network 4 times slower.

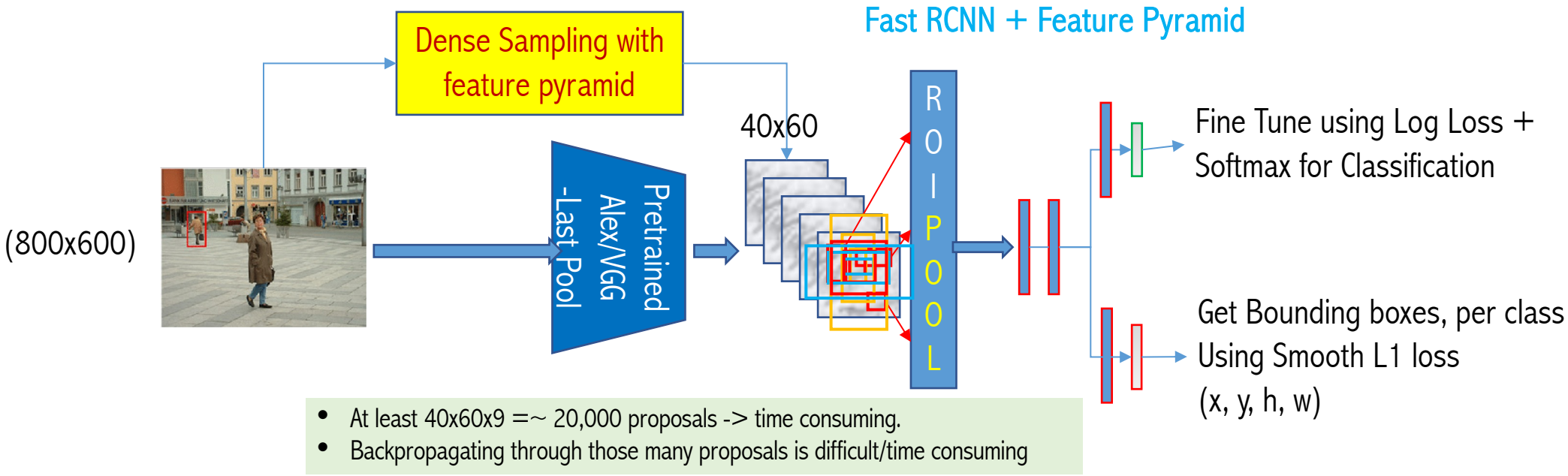


Fast RCNN + Sliding Window + Image Pyramid

Solutions for Replace Region Propsals

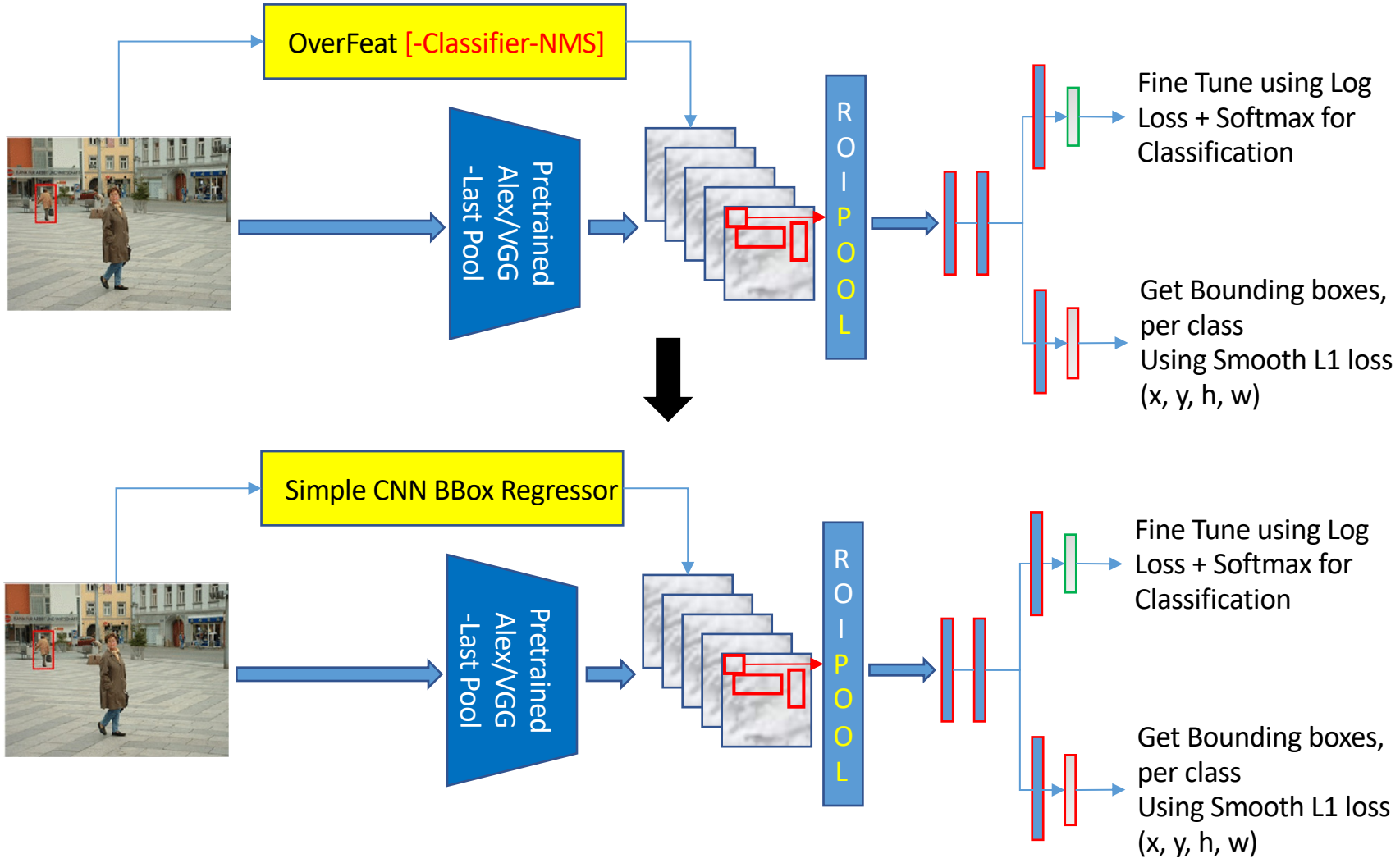
Another option is to use sliding windows of different sizes (9, as shown above) on the Feature Map. This concept is called the Feature Pyramid. This involves the use of sliding windows of 9 different sizes on top of the feature maps.





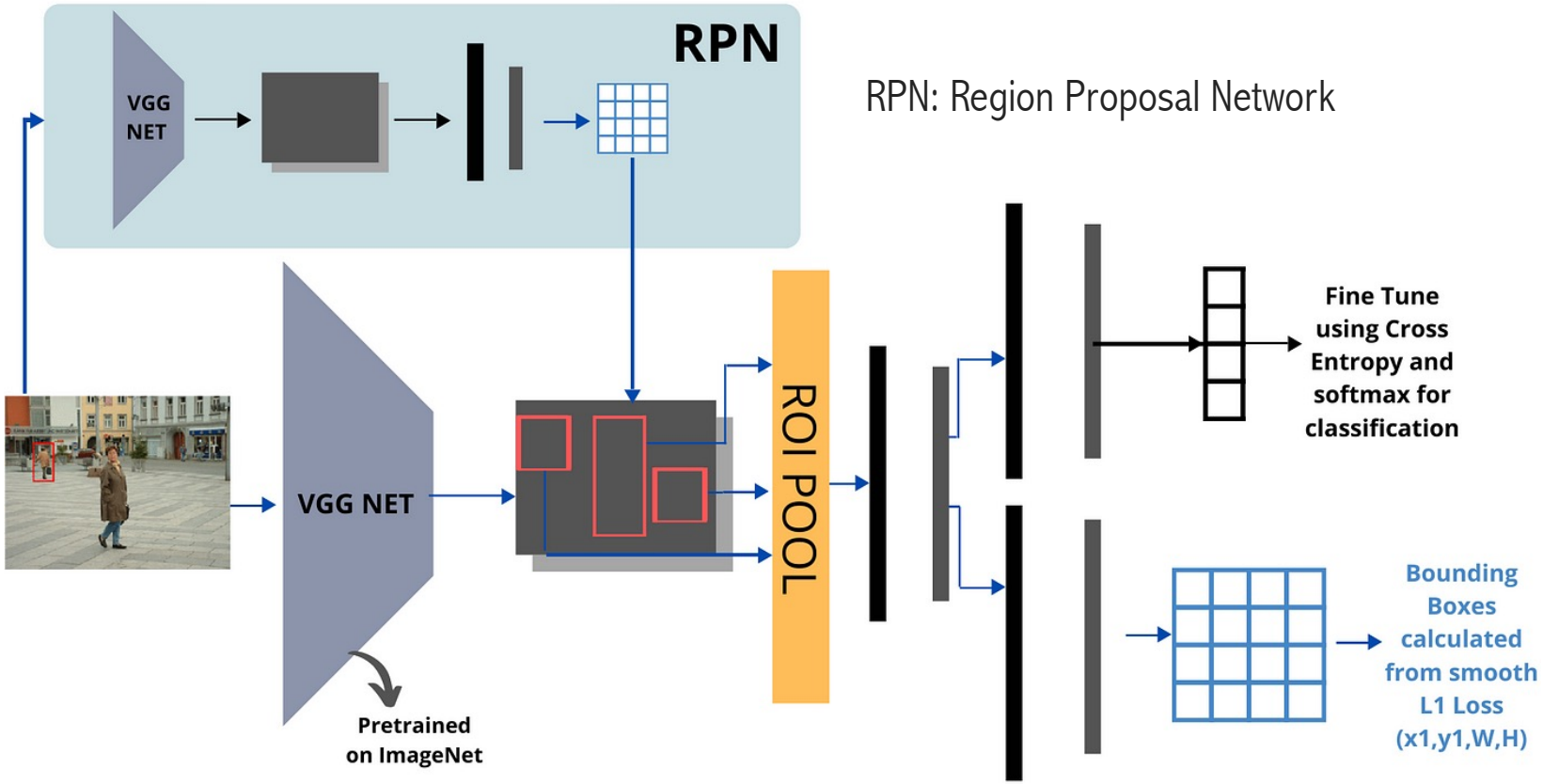
- At least 40x60x9 =~ 20,000 proposals -> time consuming.
- Backpropagating through those many proposals is difficult/time consuming

Fast RCNN + Neural Network

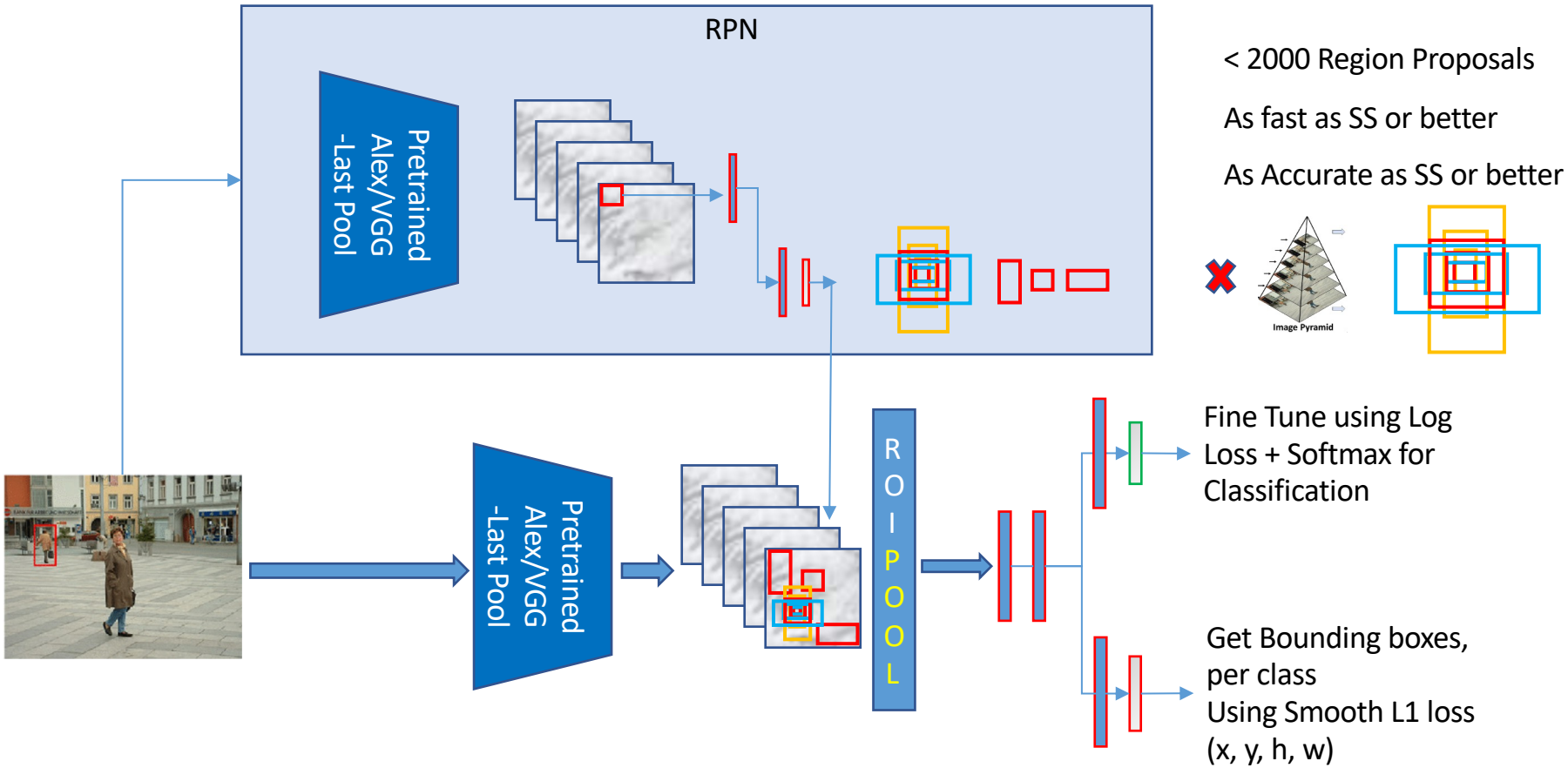


Solutions for Replace Region Propsals

Consider using a simple CNN BBox regressor in place of Selective Search to get the approximate region proposals of the image which could further be fed to the underlying Fast R-CNN architecture. This is the core idea behind Faster R-CNN.



Fast RCNN + RPN



Ideas for *Localization* using ConvNets

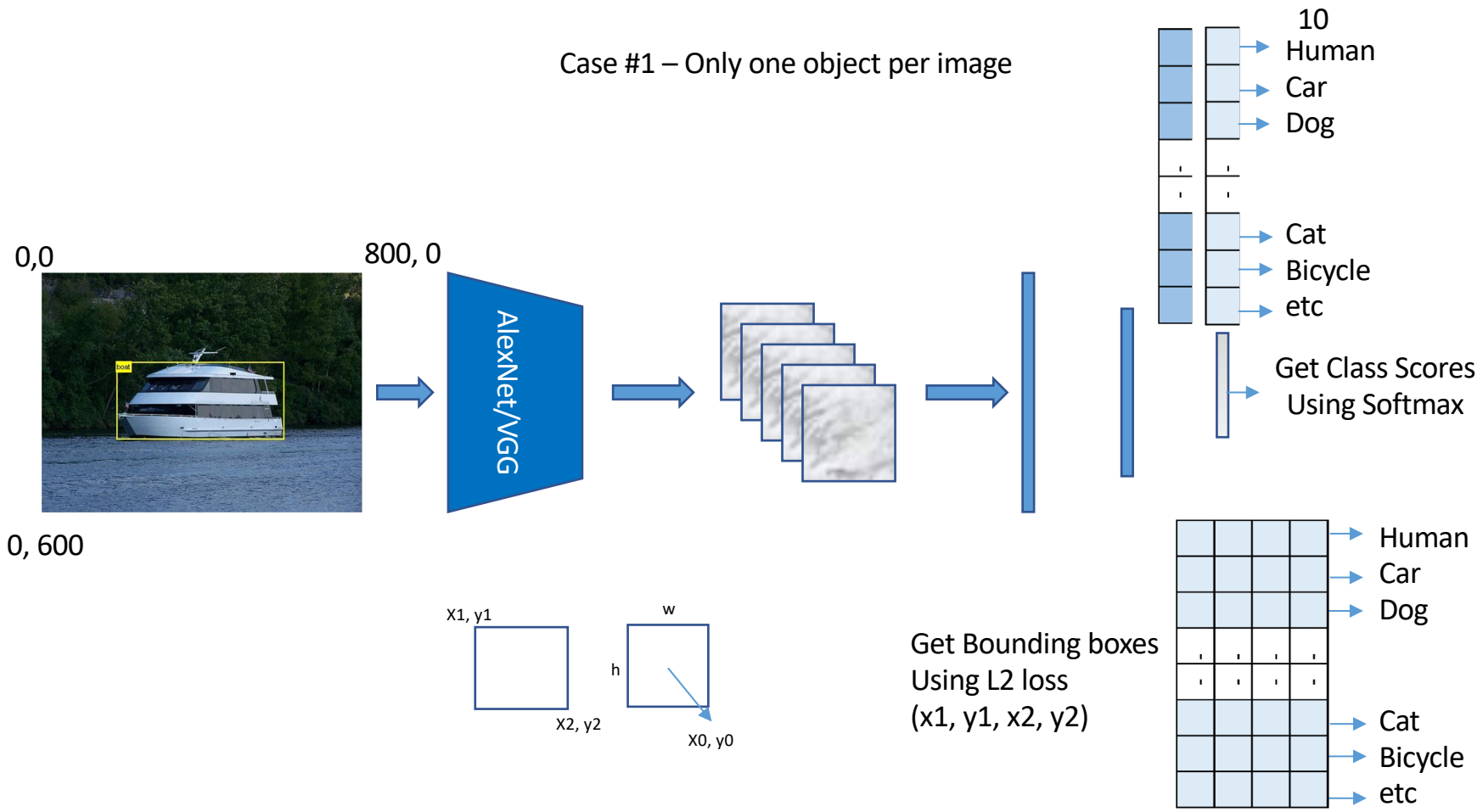
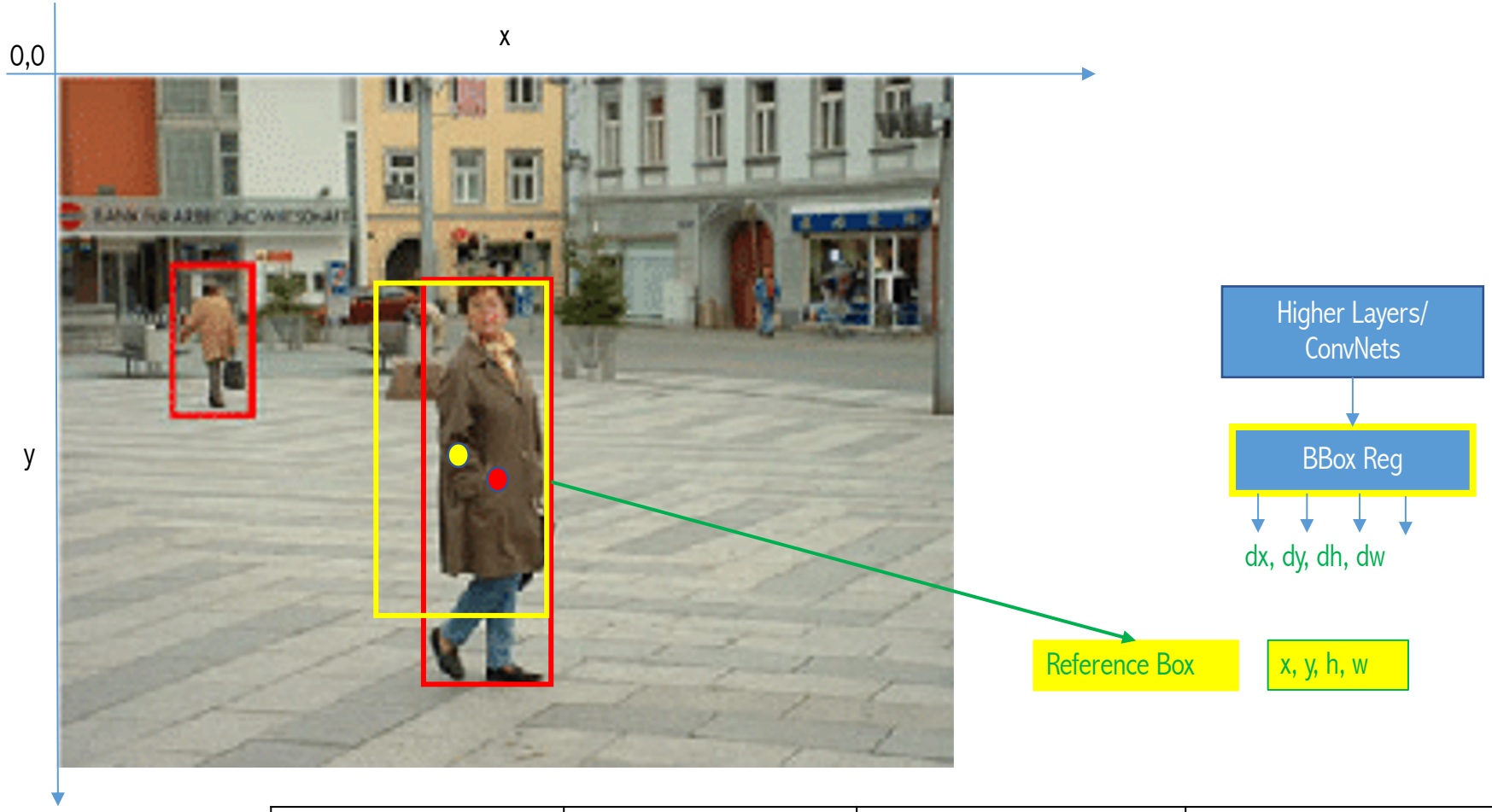


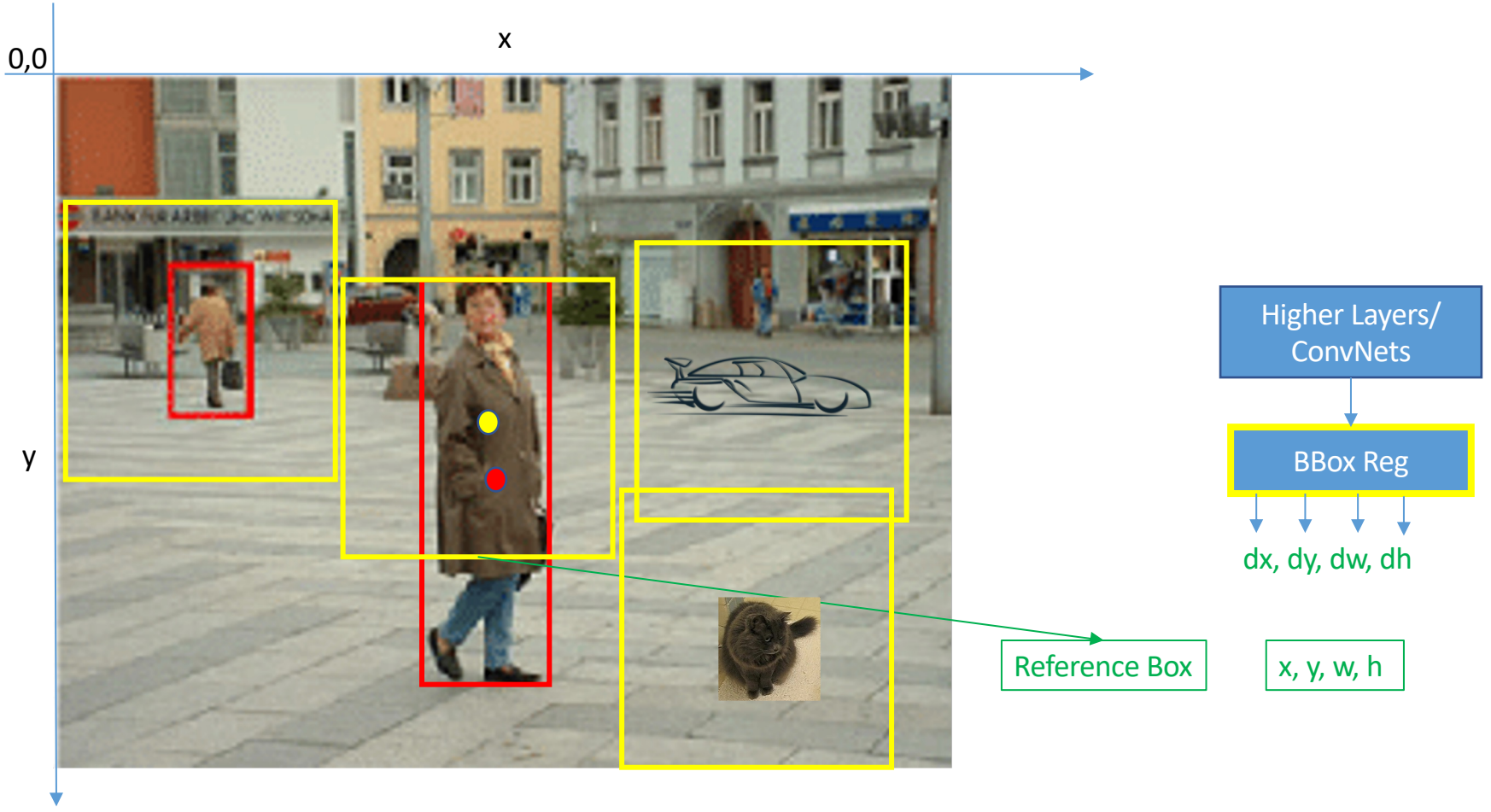
Image Credit - http://host.robots.ox.ac.uk/pascal/VOC/voc2012/examples/index.html

BBox Regression - Relative

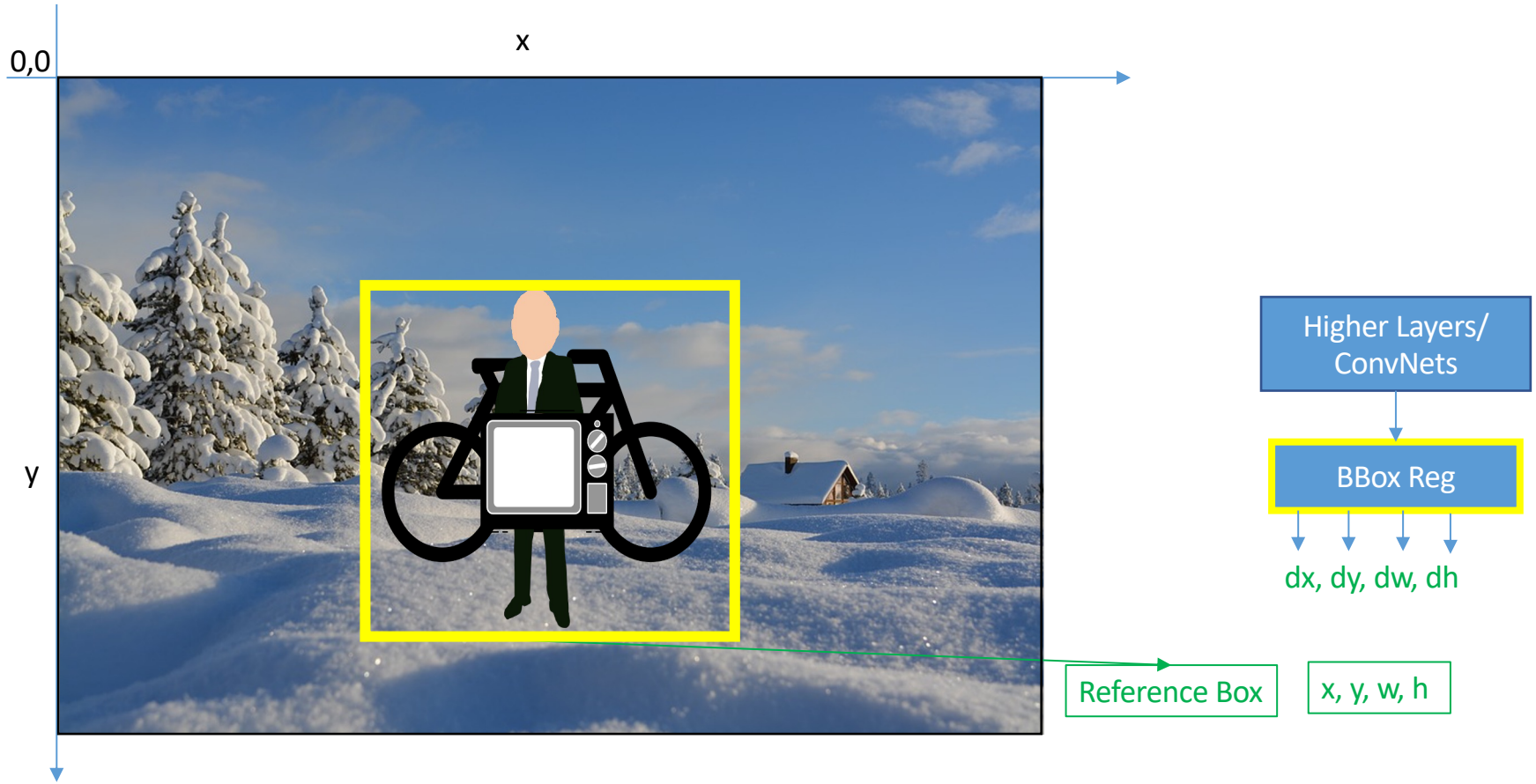


ROI reference				Bbox deltas				Predicted				Expected			
x	y	h	w	dx	dy	dh	dw	x	y	h	w	x	y	h	w
160	240	150	150	18	-22	-30	-125	178	218	120	25	180	220	120	30

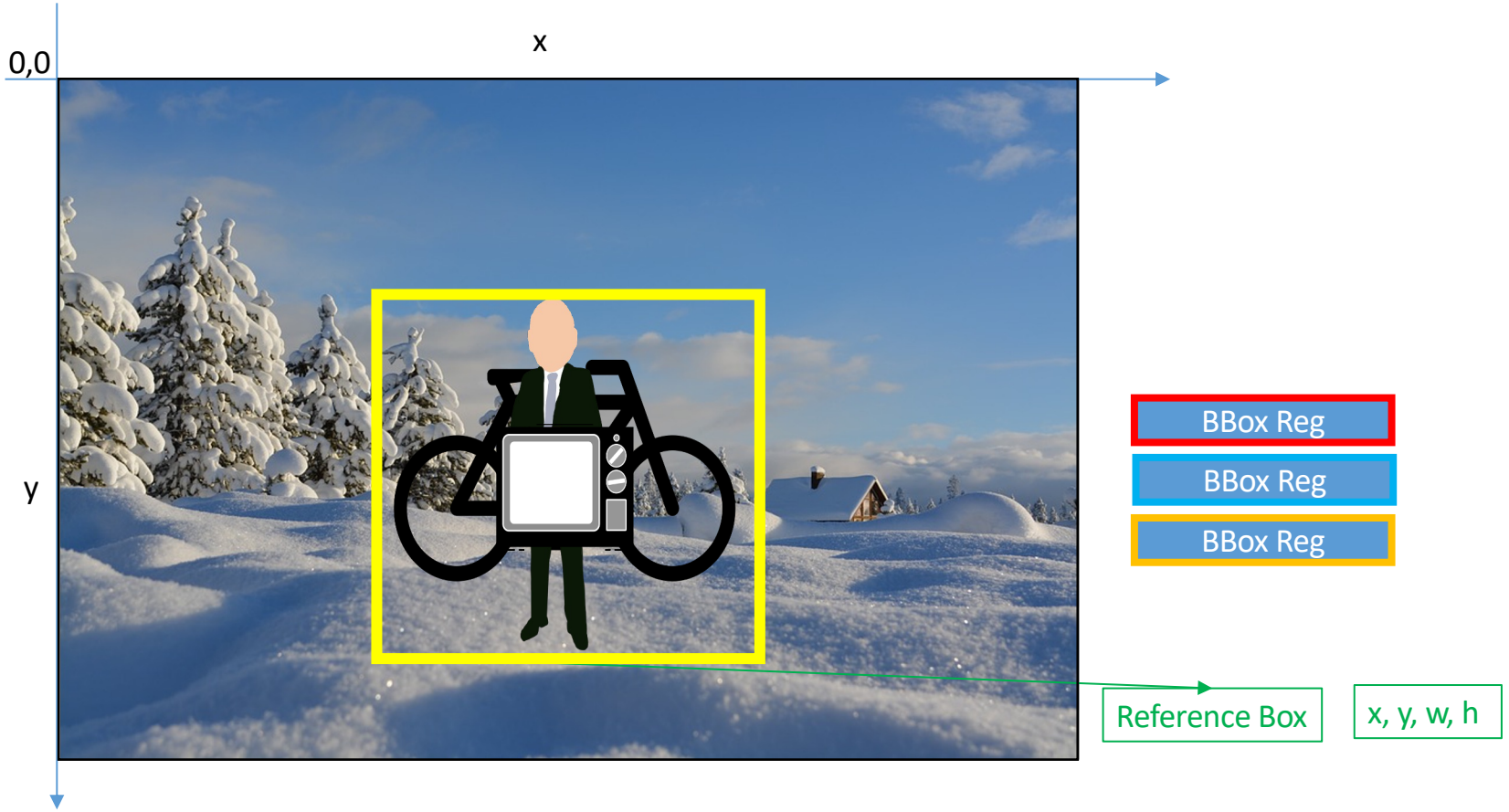
Sliding Window as Reference Box



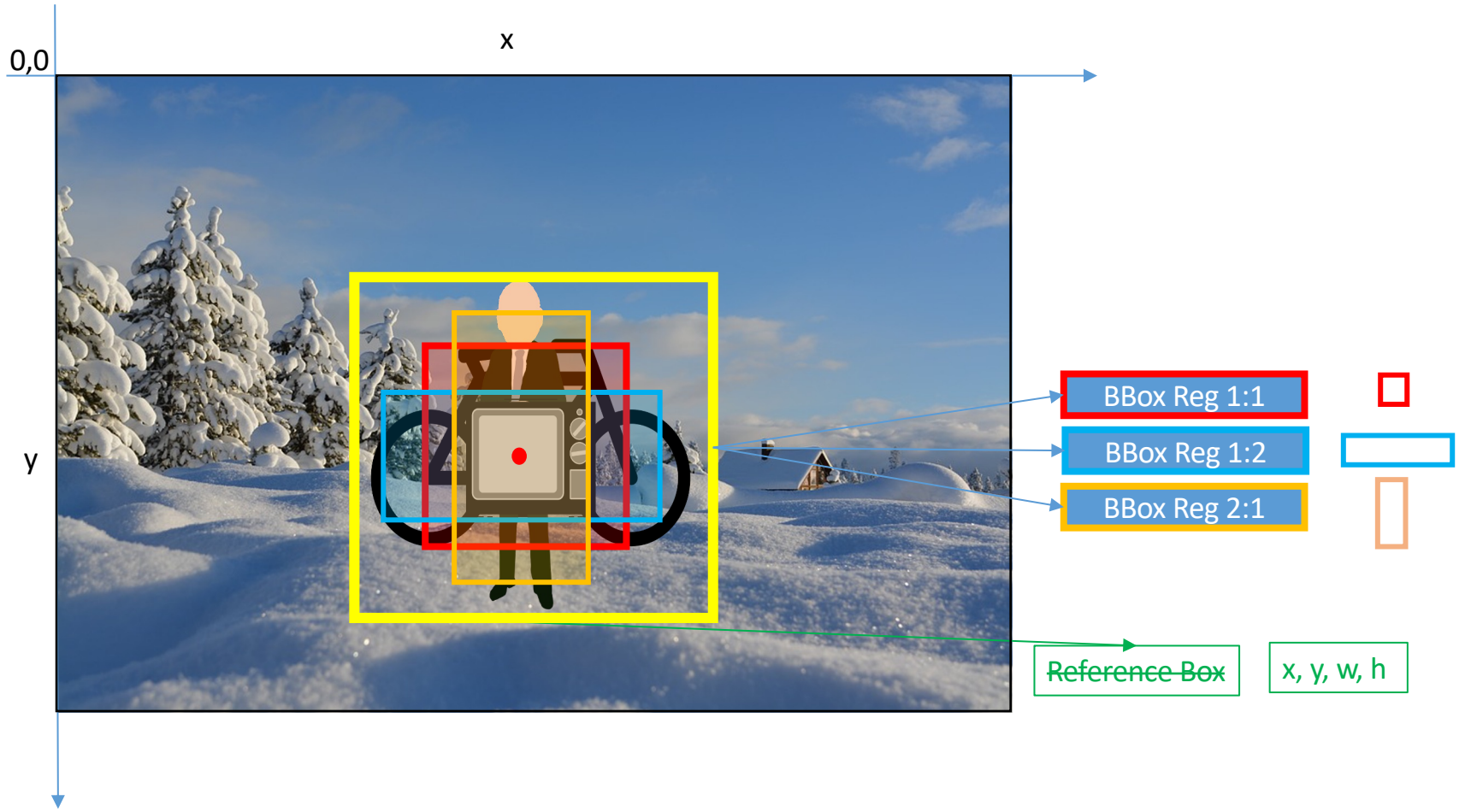
Sliding Window as Reference Box



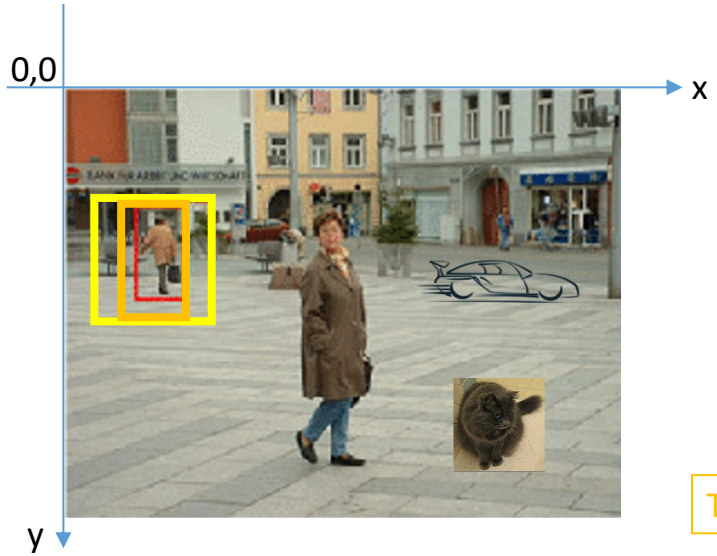
Sliding Window as Reference Box



Sliding Window as Reference Box



Square ROI to Rectangular Proposal



Higher Layers/ ConvNets

BBox Reg 2:1

dx, dy, dw, dh

ROI Centre/W/H

x, y, w, h

Ground Truth

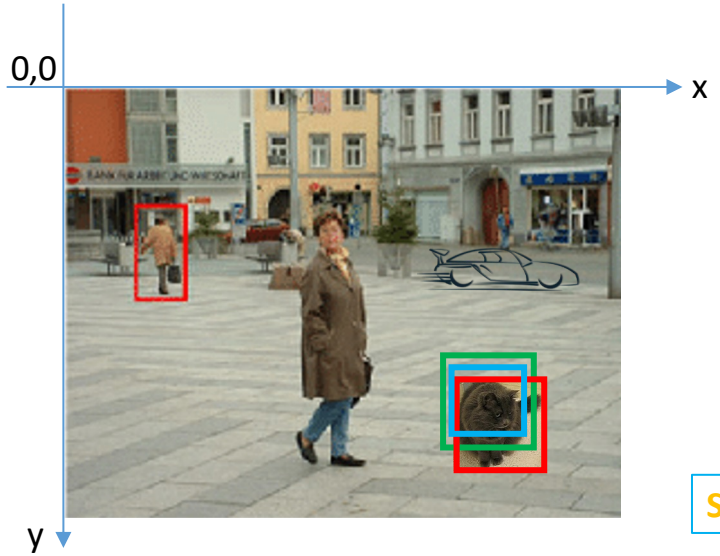
x, y, w, h

Tall Box Centre/W/H

x, y, w, h

ROI reference				Bbox deltas				Predicted				Expected			
x	y	h	w	dx	dy	dh	dw	x	y	h	w	x	y	h	w
160	240	130	65	18	-22	-10	-32	178	218	120	33	180	220	120	30

Square ROI to Rectangular Proposal



Higher Layers/ ConvNets

BBox Reg 1:1

dx, dy, dw, dh

ROI Centre/W/H

x, y, w, h

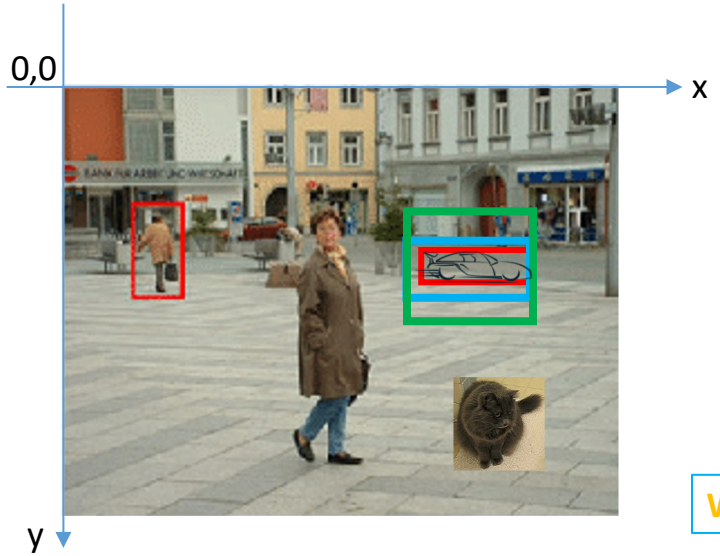
Square Box Centre/W/H

x, y, w, h

Ground Truth

x, y, w, h

Square ROI to Rectangular Proposal



ROI Centre/W/H

x, y, w, h

Wide Box Centre/W/H

x, y, w, h

Higher Layers/ ConvNets

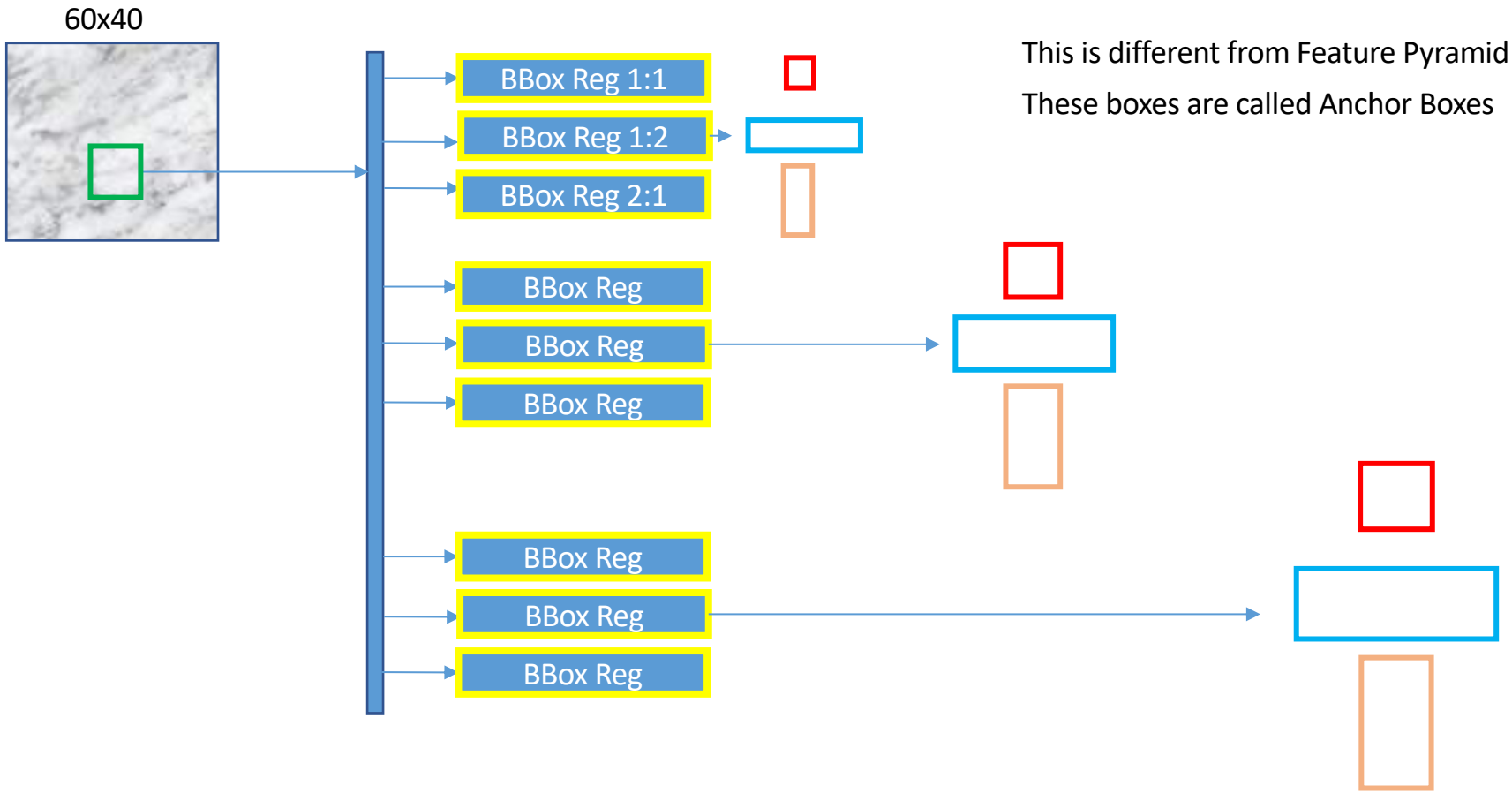
BBox Reg 1:2

dx, dy, dw, dh

Ground Truth

x, y, w, h

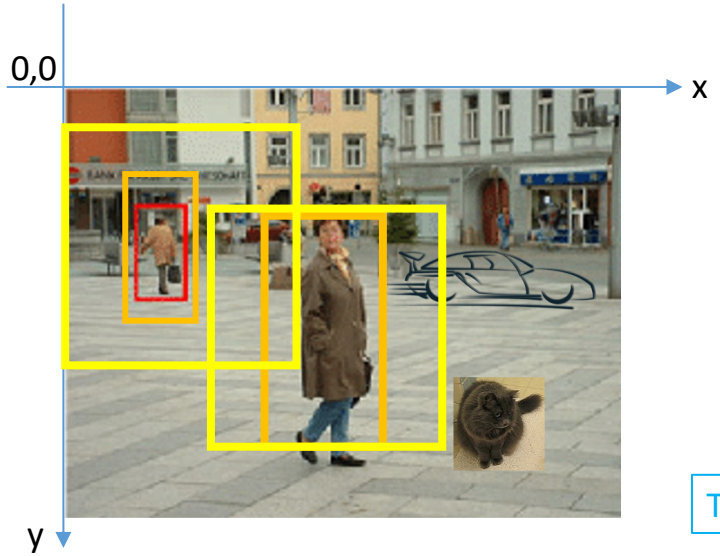
Multiple BBox Reg using Reference Boxes



This is different from Feature Pyramid

These boxes are called Anchor Boxes

Bigger Objects?



SW Centre/W/H

x, y, w, h

Tall Box Centre/W/H

x, y, w, h

Higher Layers/ ConvNets

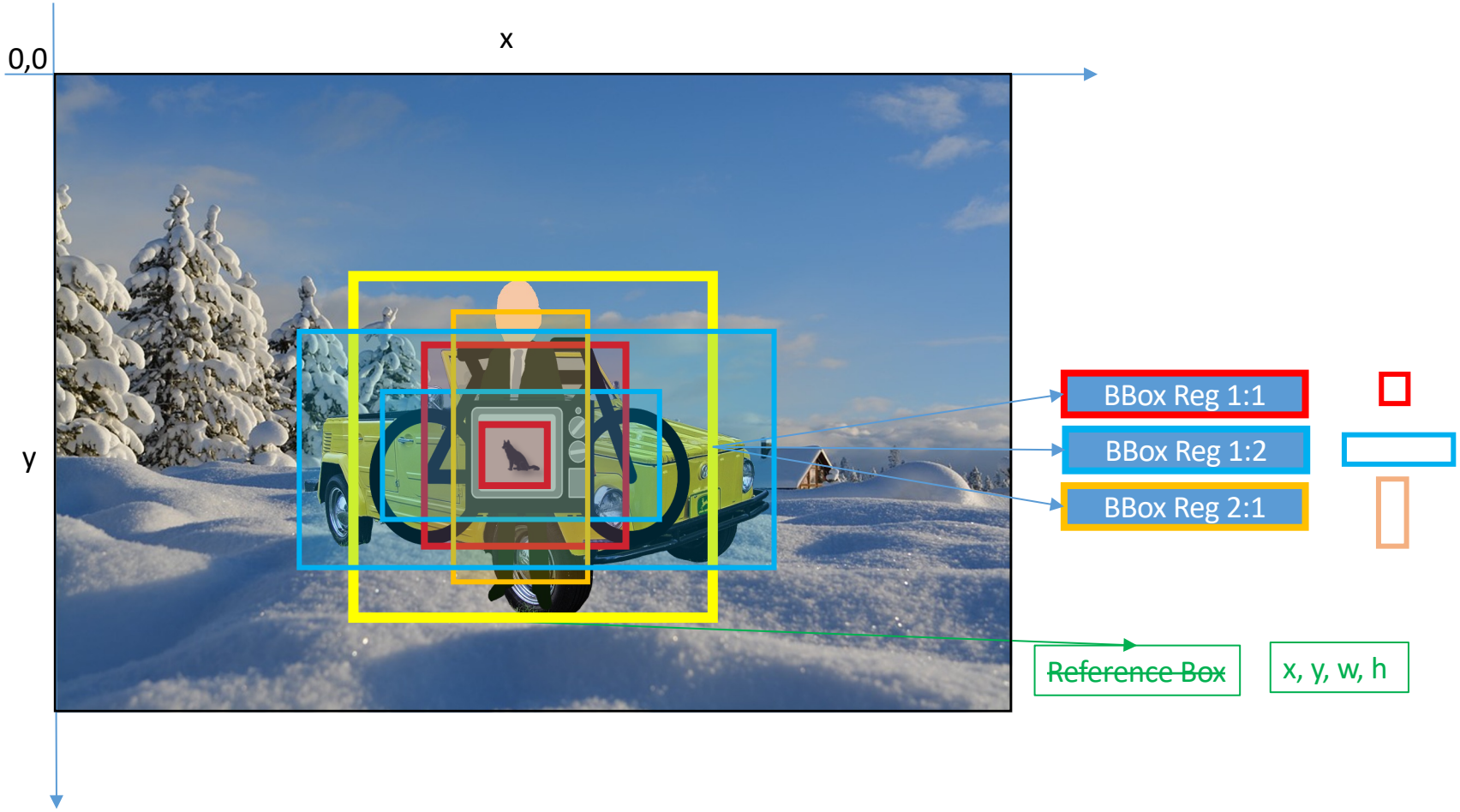
BBox Reg

dx, dy, dw, dh

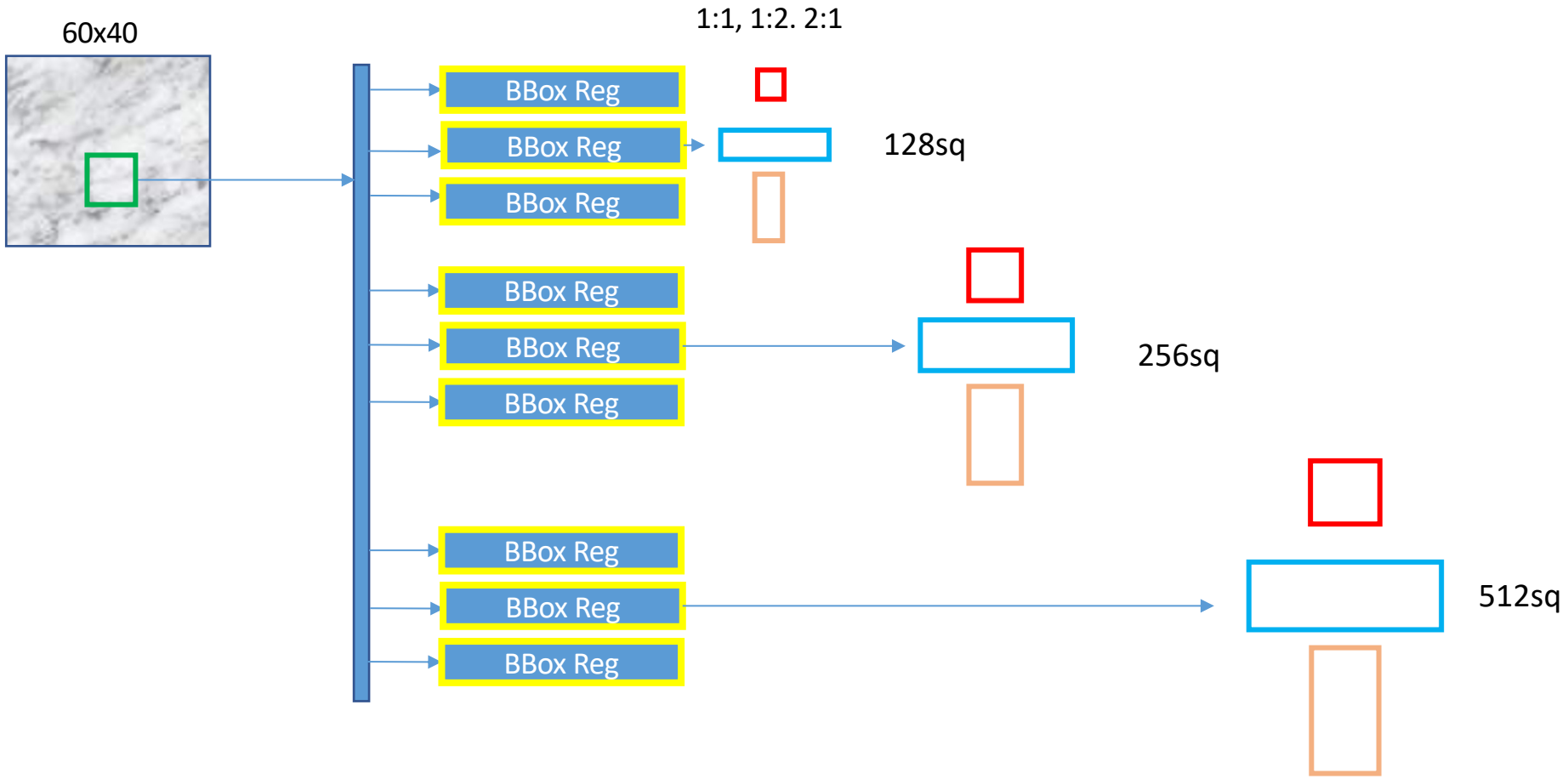
Ground Truth

x, y, w, h

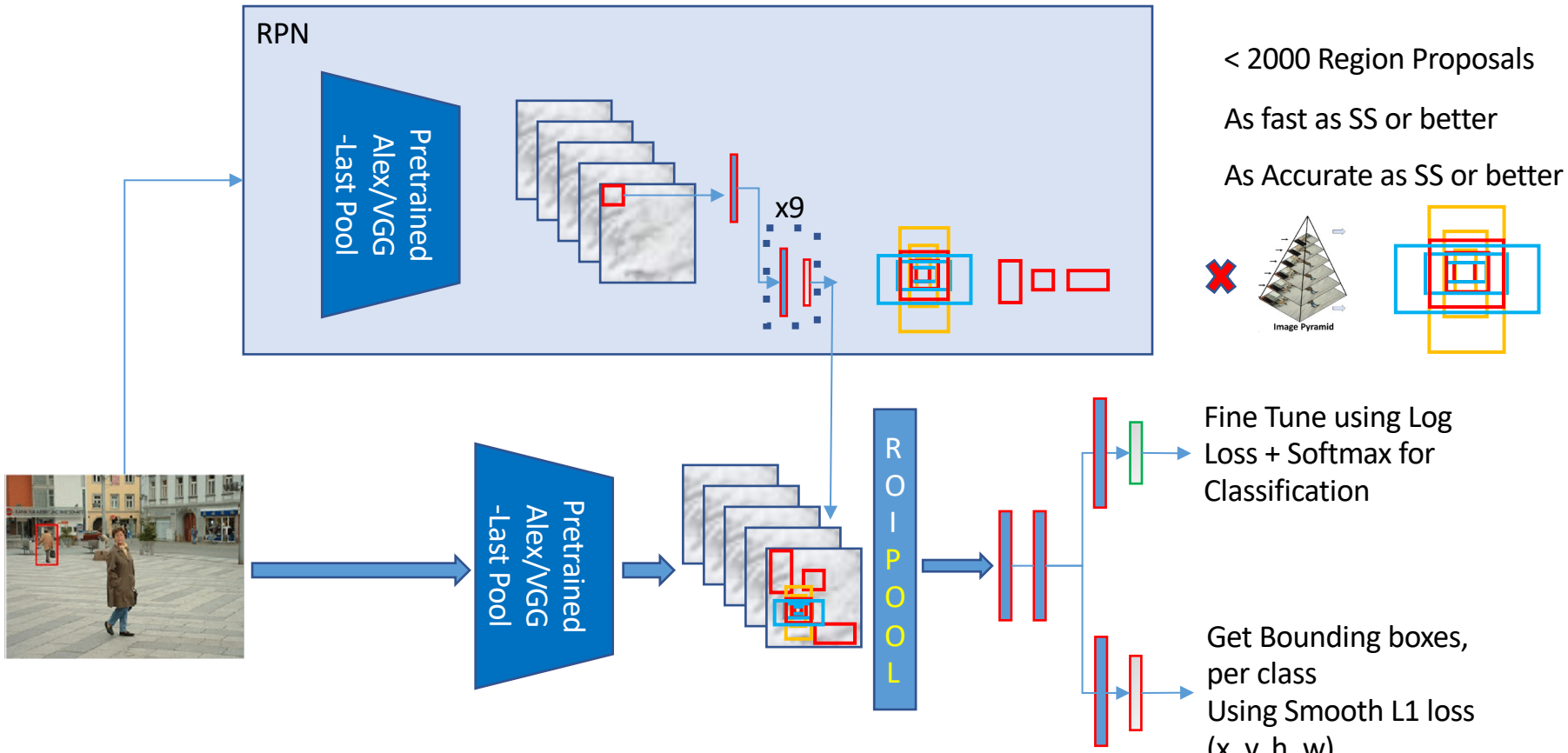
Bigger Objects?



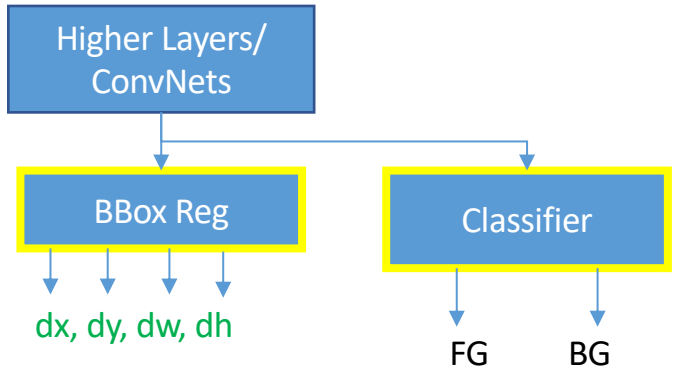
Multiple BBox Reg using Anchor Boxes



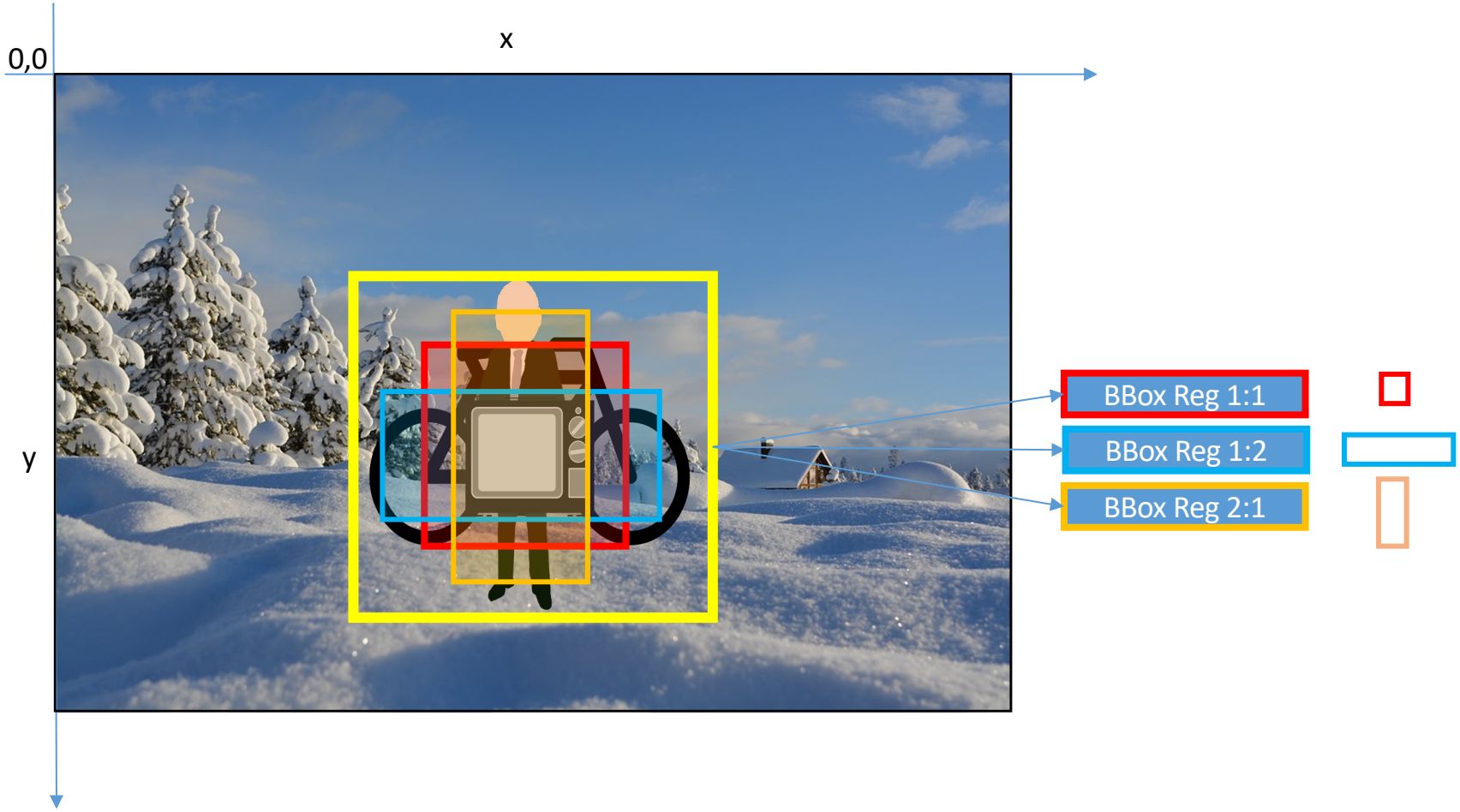
Fast RCNN + RPN



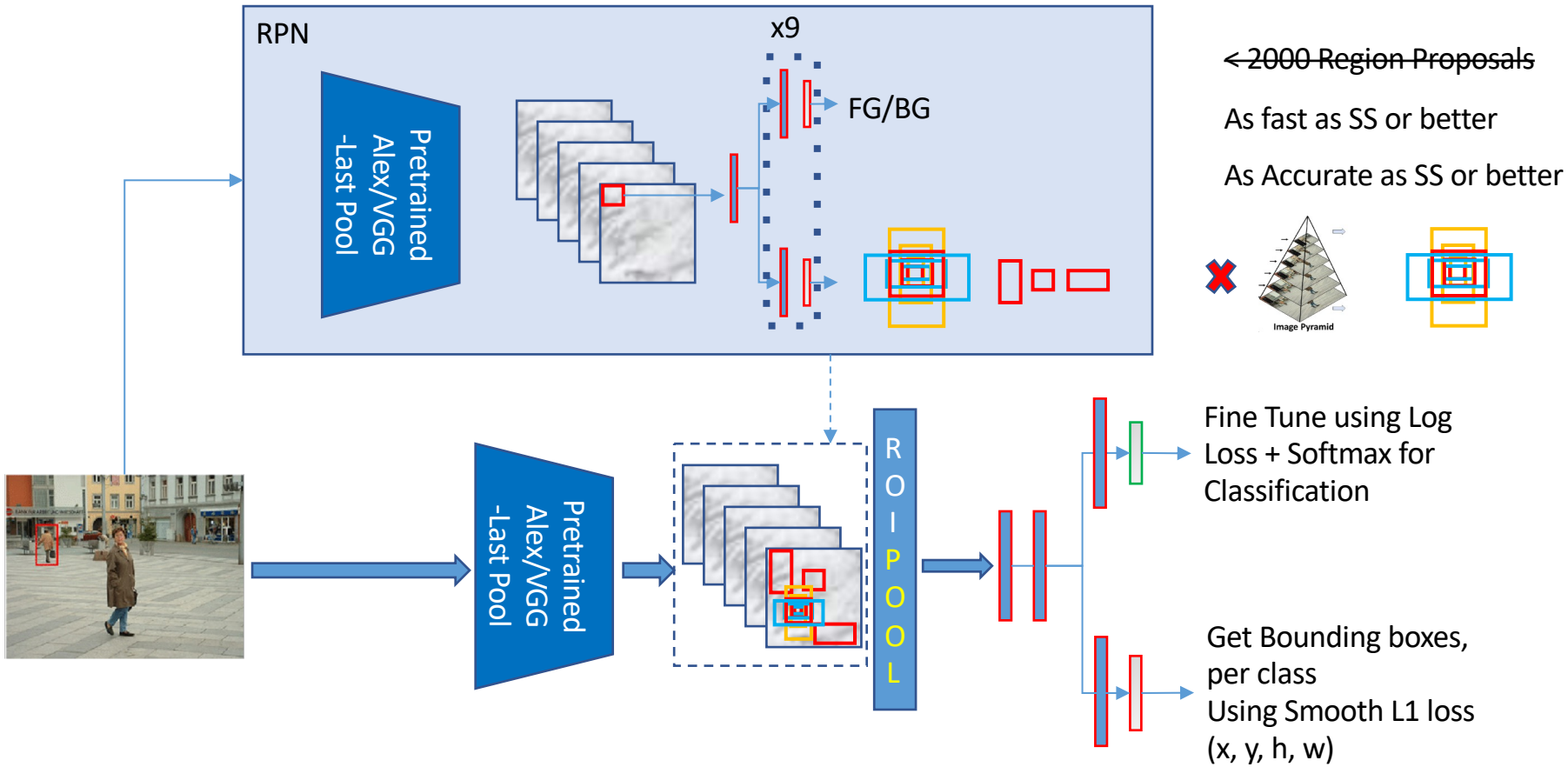
How to reduce number of proposals?



Faster RCNN – Training Anchor Boxes - Labelling



Fast RCNN + RPN

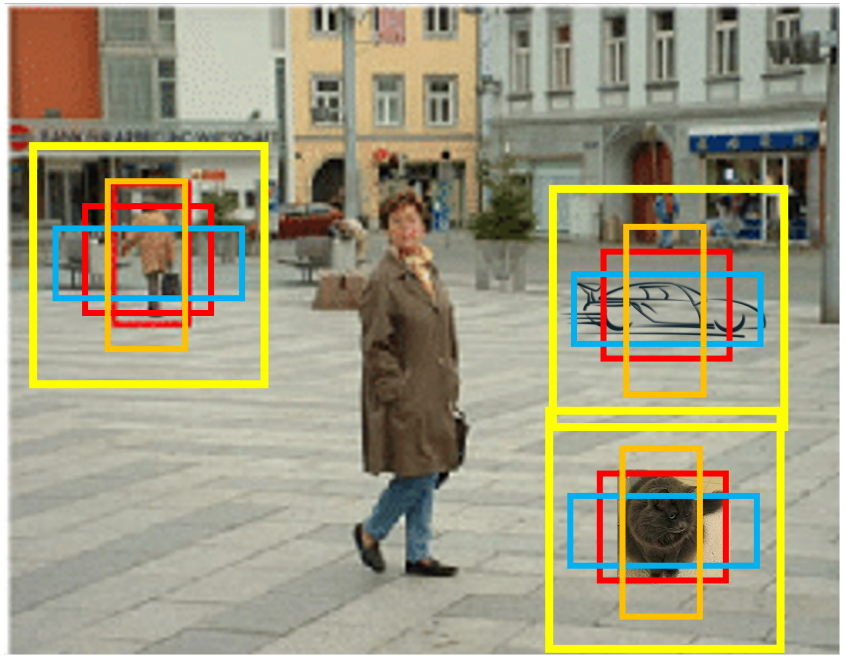


Faster RCNN – Training Anchor Boxes - Labelling

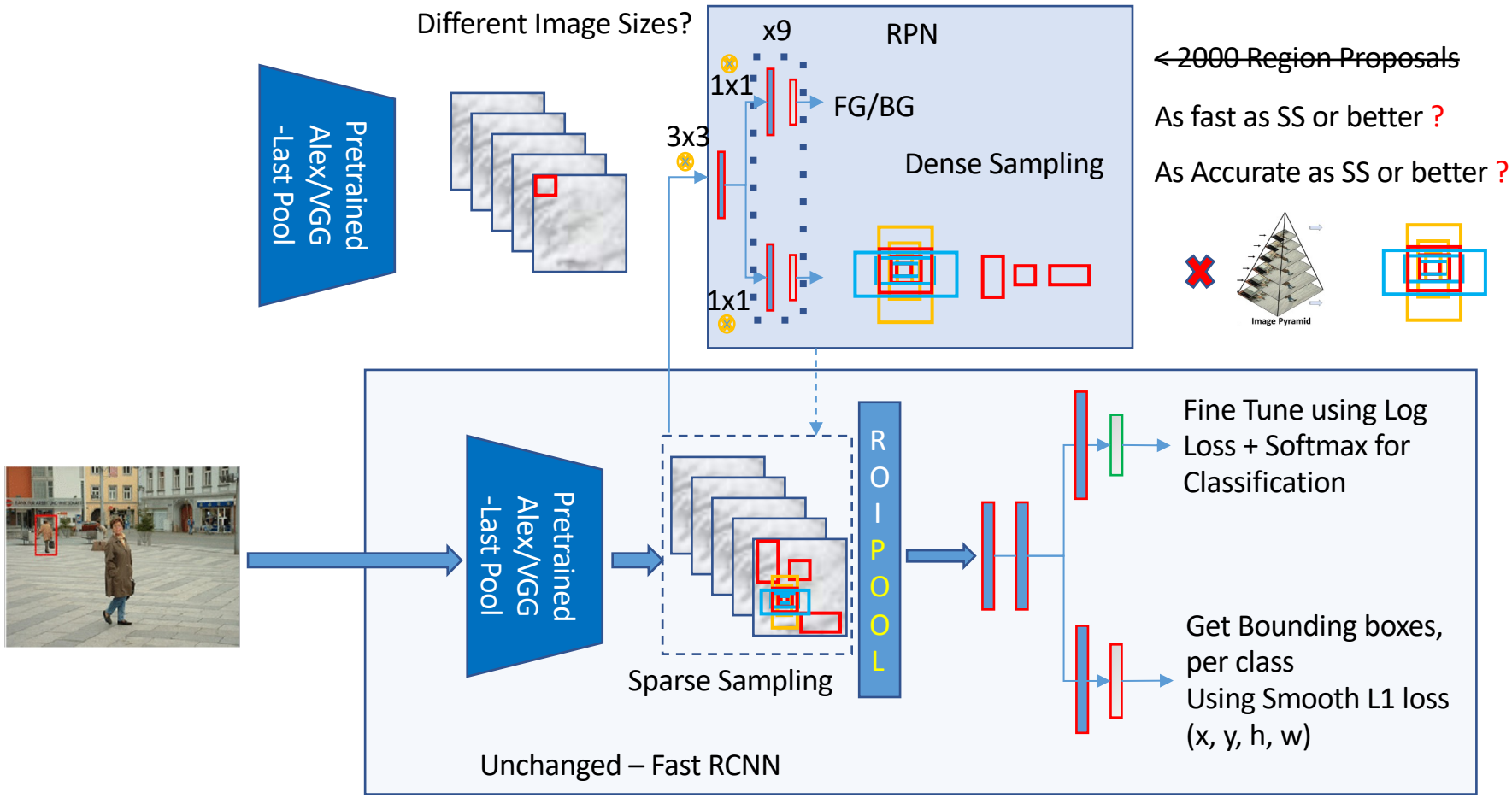
BBox Reg 1:1

BBox Reg 1:2

BBox Reg 2:1



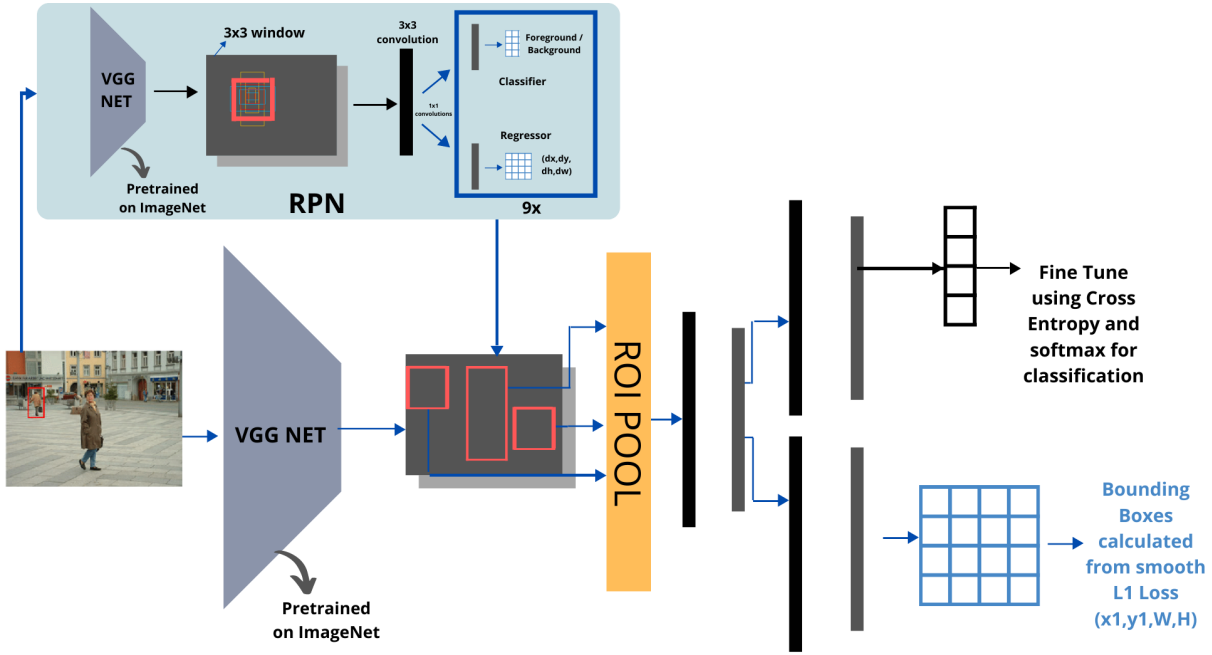
Fast RCNN + RPN = Faster RCNN



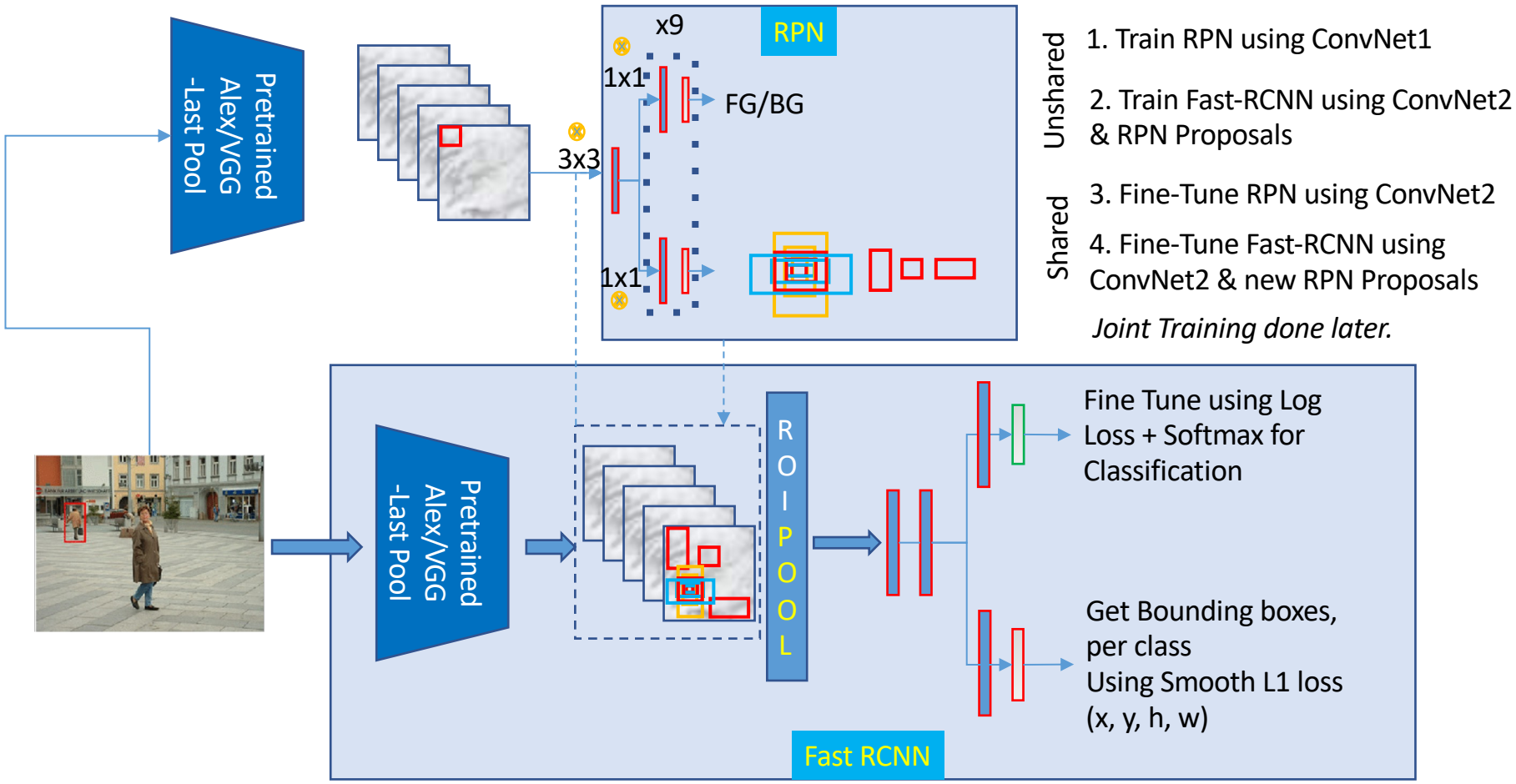
Faster & Accurate than SS?

Time in ms

model	system	conv	proposal	region-wise	total	rate	mAP
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps	66.9
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**	69.9



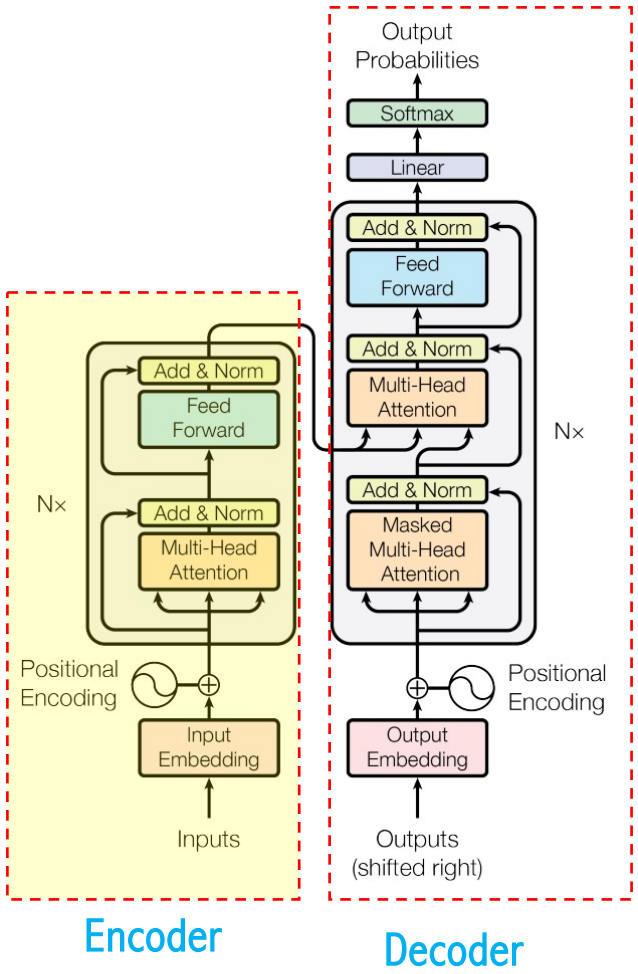
Faster RCNN - Training



Outline

- CNN Limitations
- Region Based Convolutional Neural Networks
- Spatial Pyramid Pooling
- Fast R-CNN
- Faster RCNN
- Object Detection with Transformers: Motivation

Transformer Motivation for OD



Input

Output

English → Vietnamese

I watch movie → Tôi xem phim

Word embeddeding

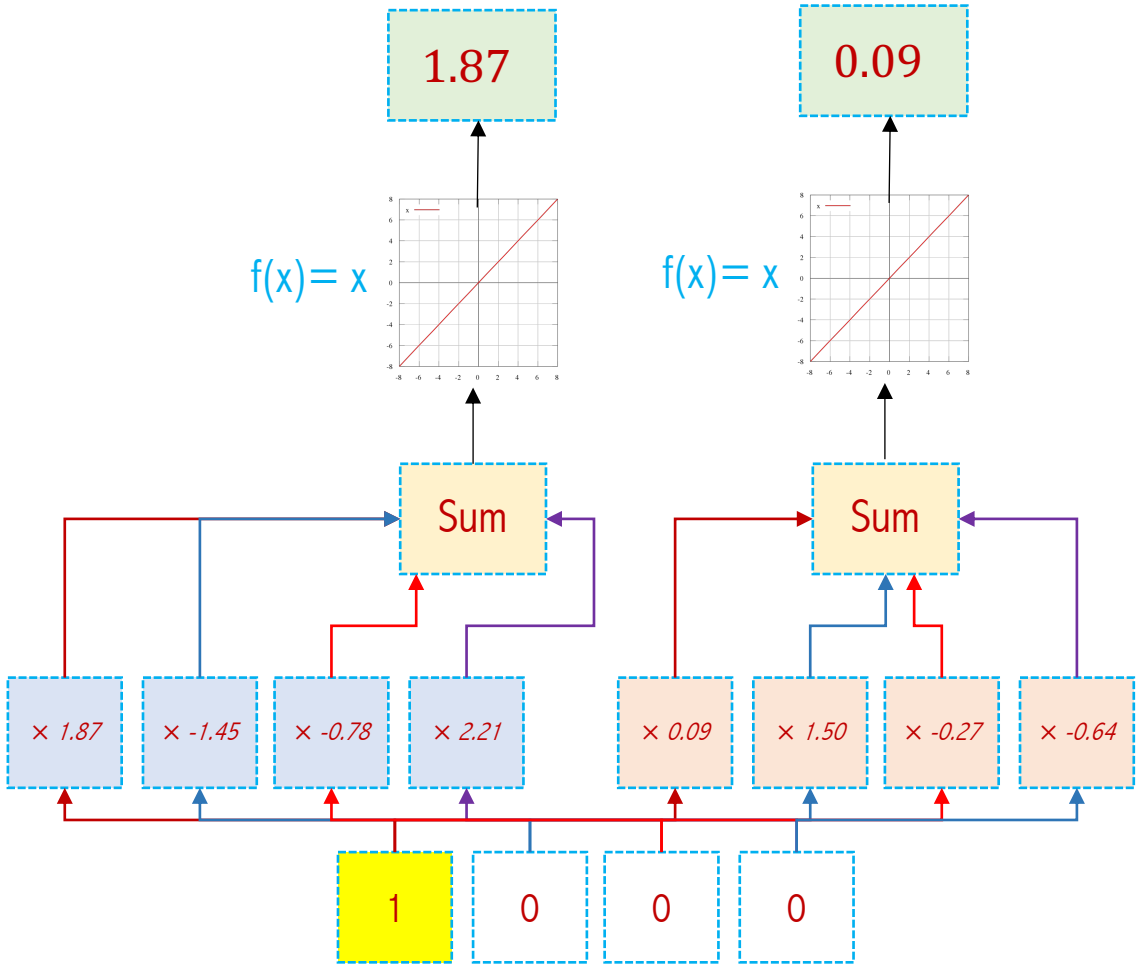
1.9 0.1
-1.5 1.5
1.6 -1.6

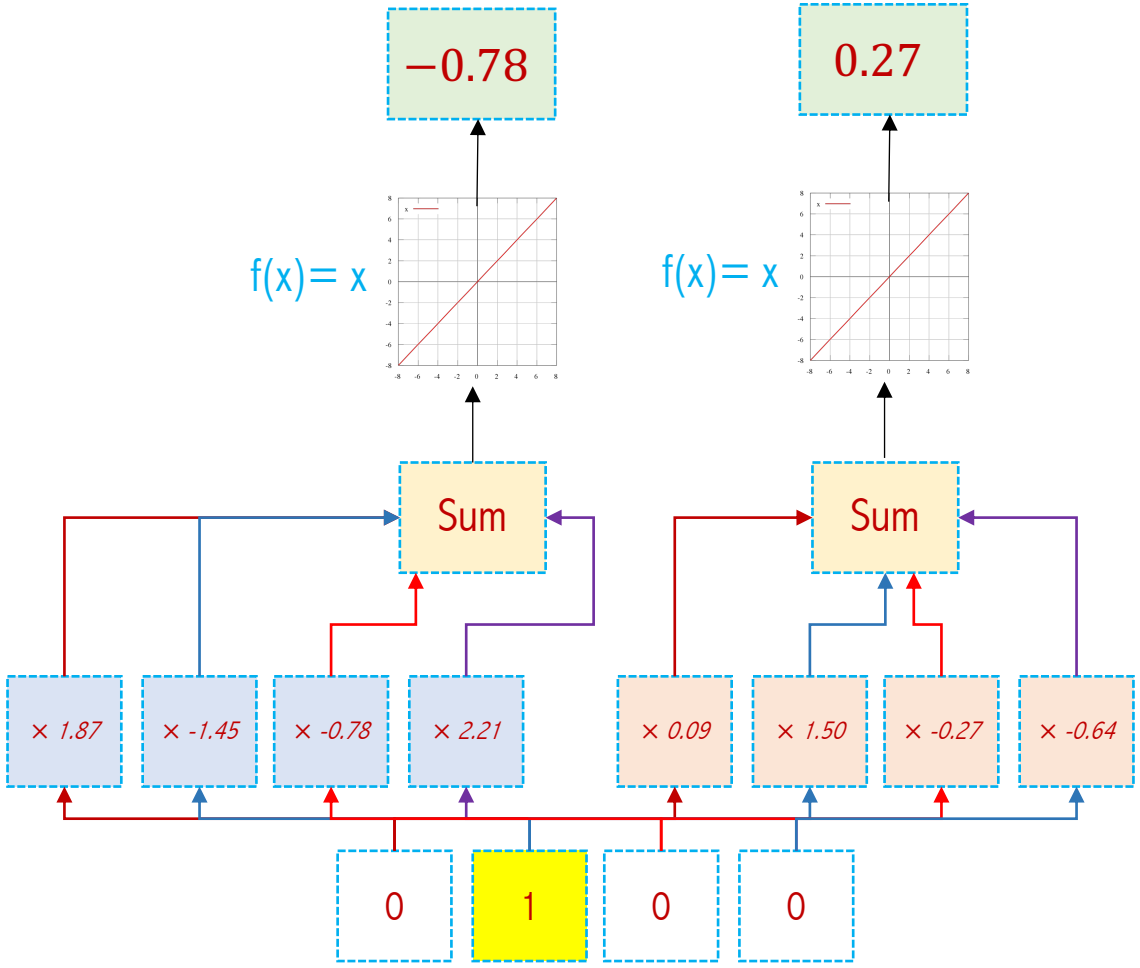
→

0.0 2.0
1.2 1.2
1.7 1.8

Transformer Encoder

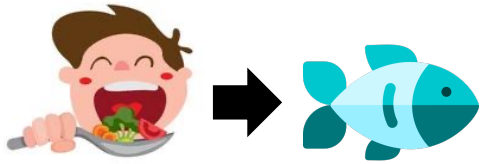
Word Embedding Using Neural Network



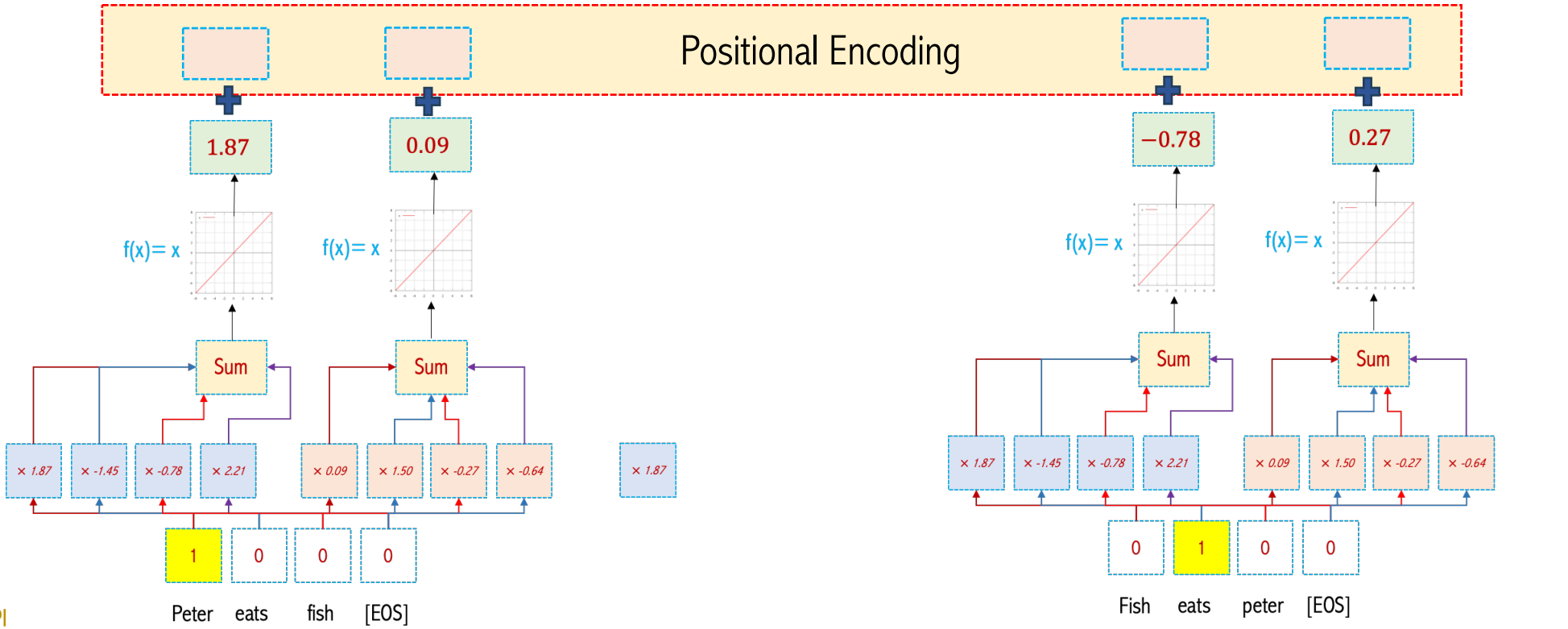


Word Ordering

Peter eats fish

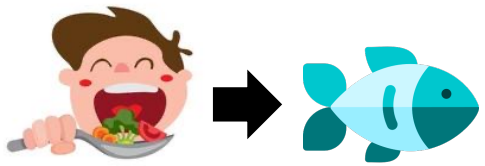


Fish eats Vinh



Word Ordering

Peter eats fish



$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

$$P(k, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$

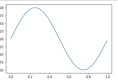
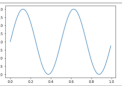
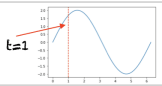
```python
import numpy as np
import matplotlib.pyplot as plt

def getPositionEncoding(seq_len, d, n=10000):
    P = np.zeros((seq_len, d))
    for k in range(seq_len):
        for i in np.arange(int(d/2)):
            denominator = np.power(n, 2*i/d)
            P[k, 2*i] = np.sin(k/denominator)
            P[k, 2*i+1] = np.cos(k/denominator)
    return P

P = getPositionEncoding(seq_len=4, d=4, n=100)
print(P)
```

Sequence	Index of token	Positional Encoding Matrix			
I	0	P_{00}	P_{01}	…	P_{0d}
am	1	P_{10}	P_{11}	…	P_{1d}
a	2	P_{20}	P_{21}	…	P_{2d}
Robot	3	P_{30}	P_{31}	…	P_{3d}

Positional Encoding Matrix for the sequence 'I am a robot'

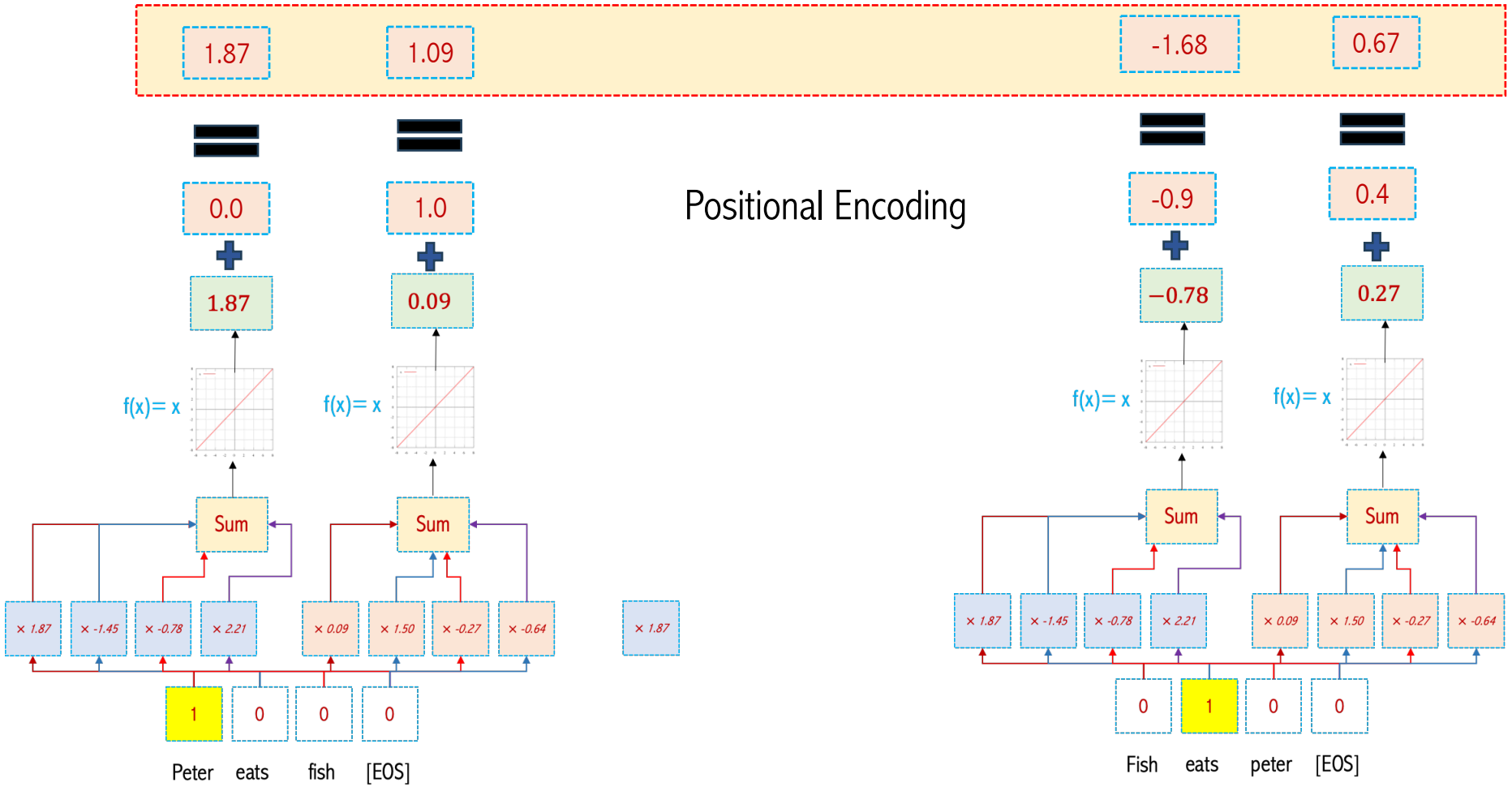
Equation	Graph	Frequency	Wavelength
$\sin(2\pi t)$		1	1
$\sin(2 * 2\pi t)$		2	1/2
$\sin(t)$		$1/2\pi$	2π
$\sin(ct)$	Depends on c	$c/2\pi$	$2\pi/c$

Sequence	Index of token, k	Positional Encoding Matrix with d=4, n=100			
		i=0	i=0	i=1	i=1
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

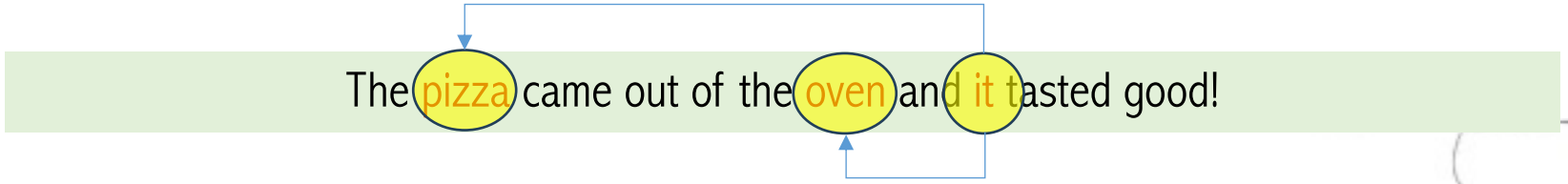
Positional Encoding Matrix for the sequence 'I am a robot'

https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

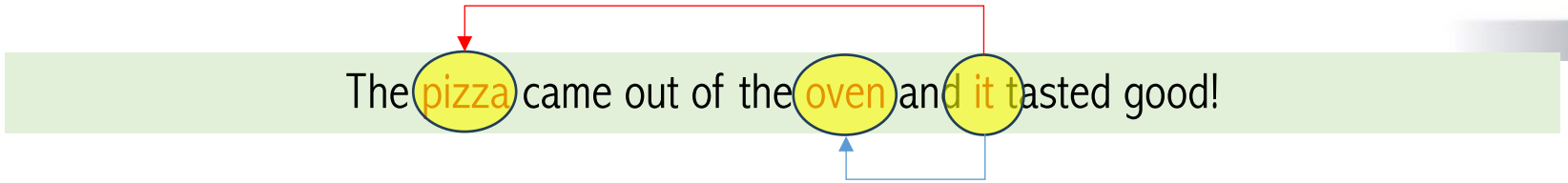
Word Ordering



Relationship Among Words: Idea

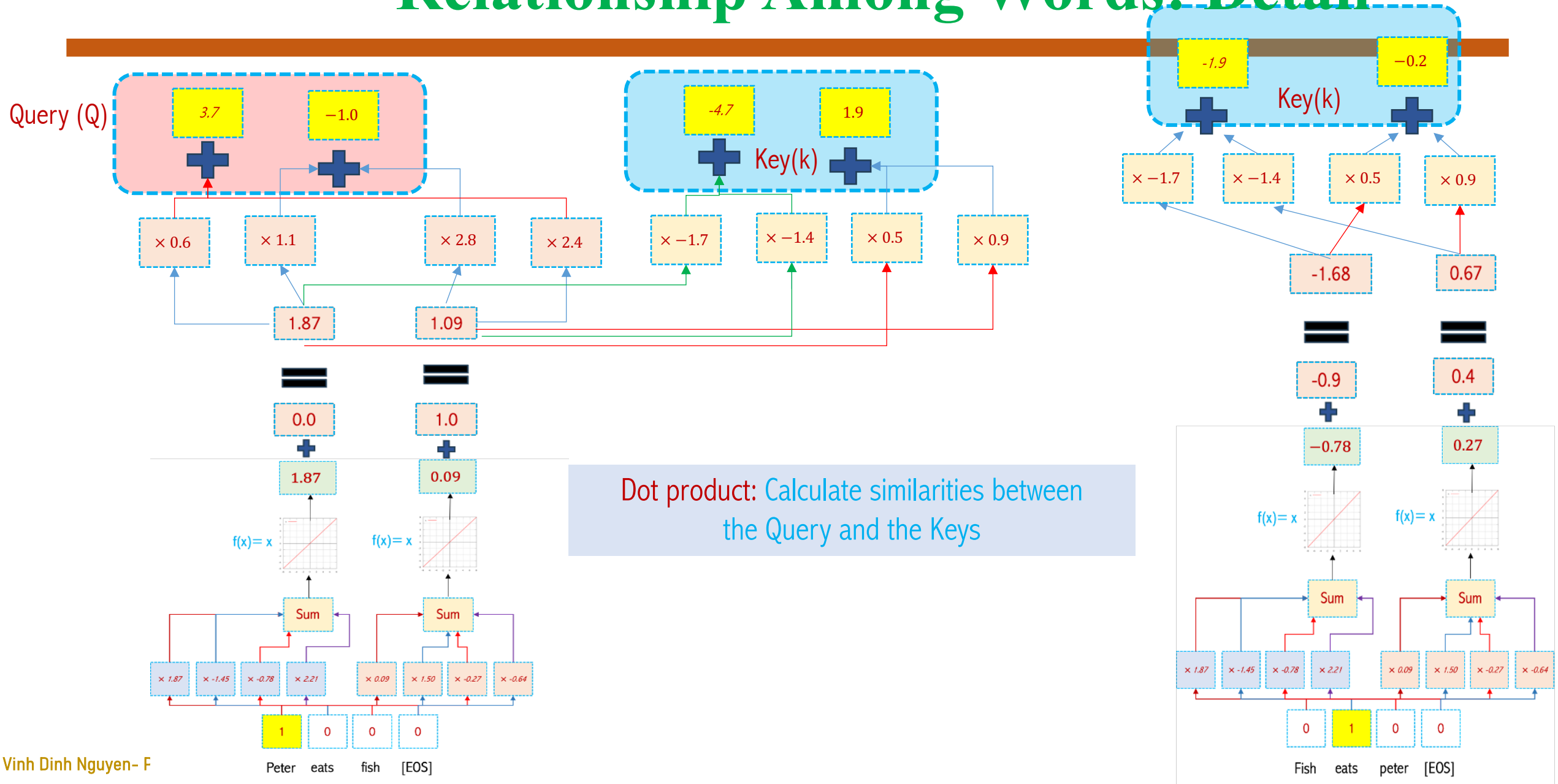


Seft-attention works by seeing how similar each word is to all of the words in the sentense, including itself



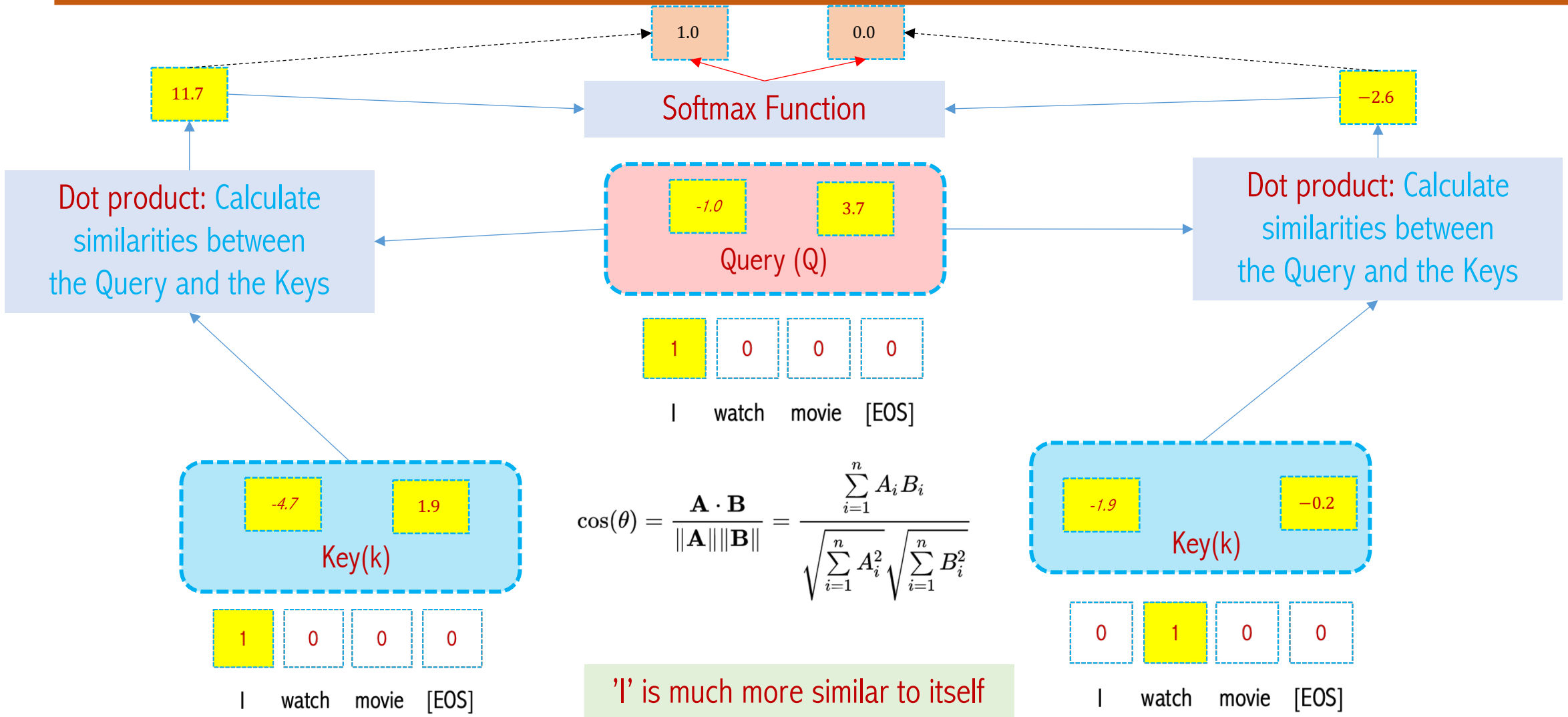
If we looked at a lot of sentenses abut pizza and the word it was more commonly associated with pizza than oven

Relationship Among Words: Detail

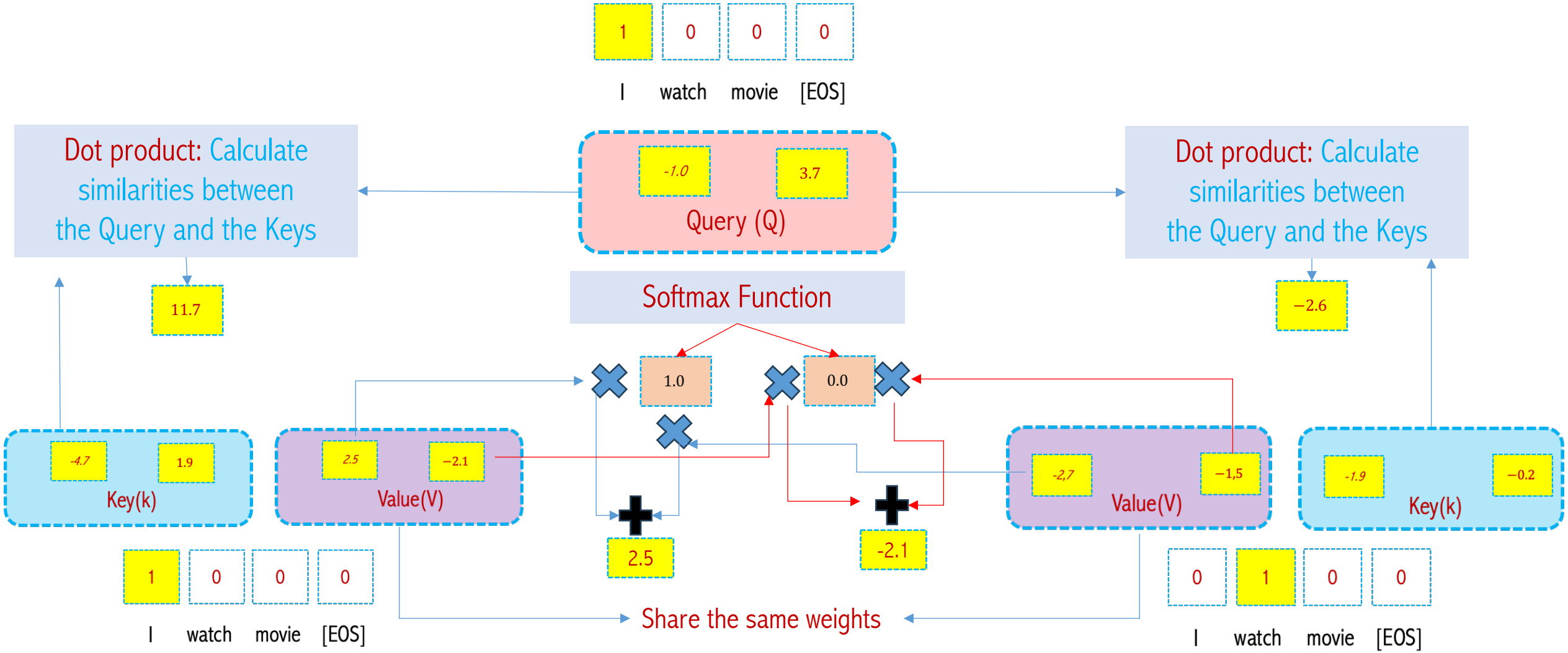


Dot product: Calculate similarities between the Query and the Keys

Relationship Among Words: Detail

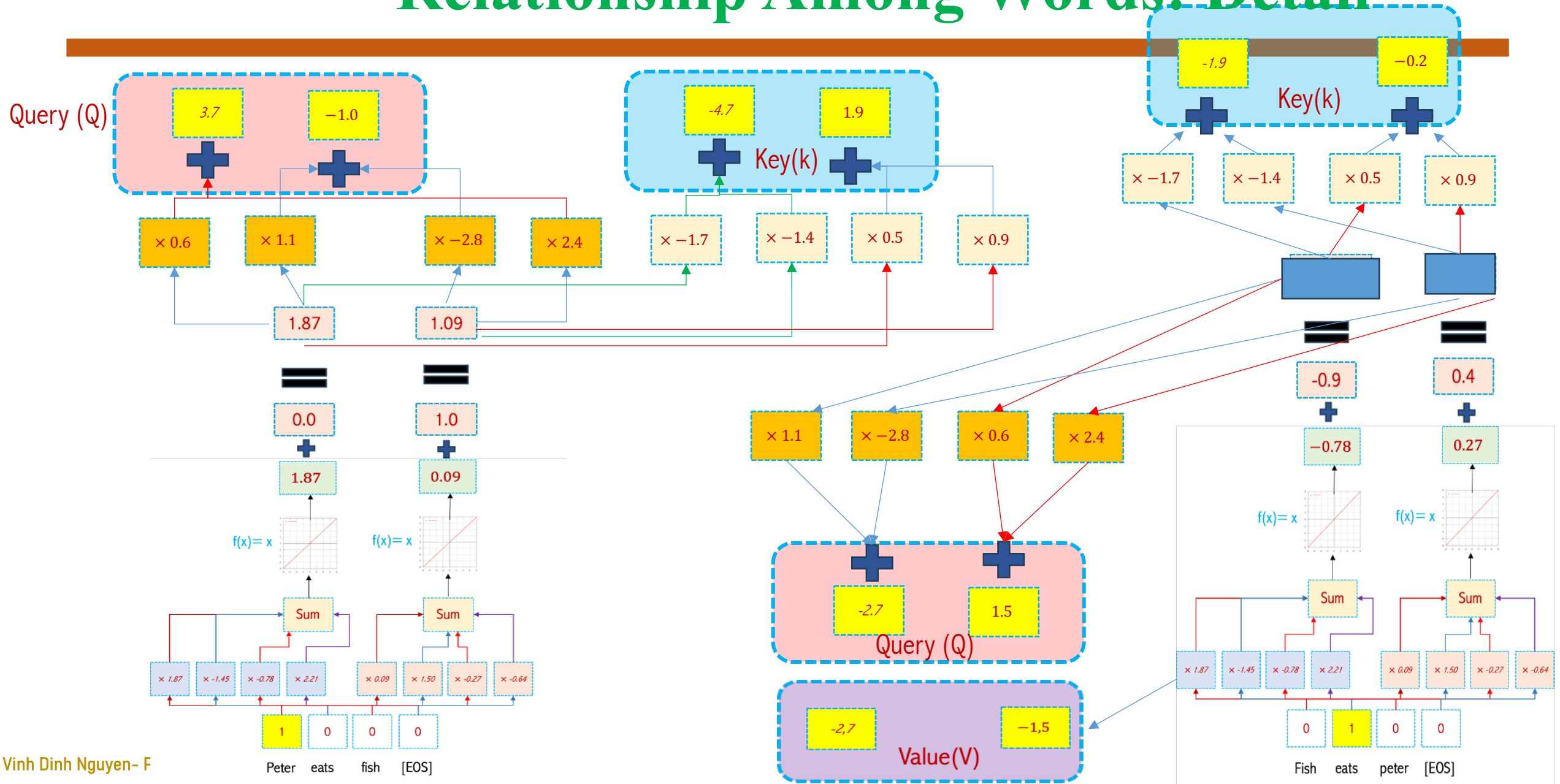


Relationship Among Words: Detail

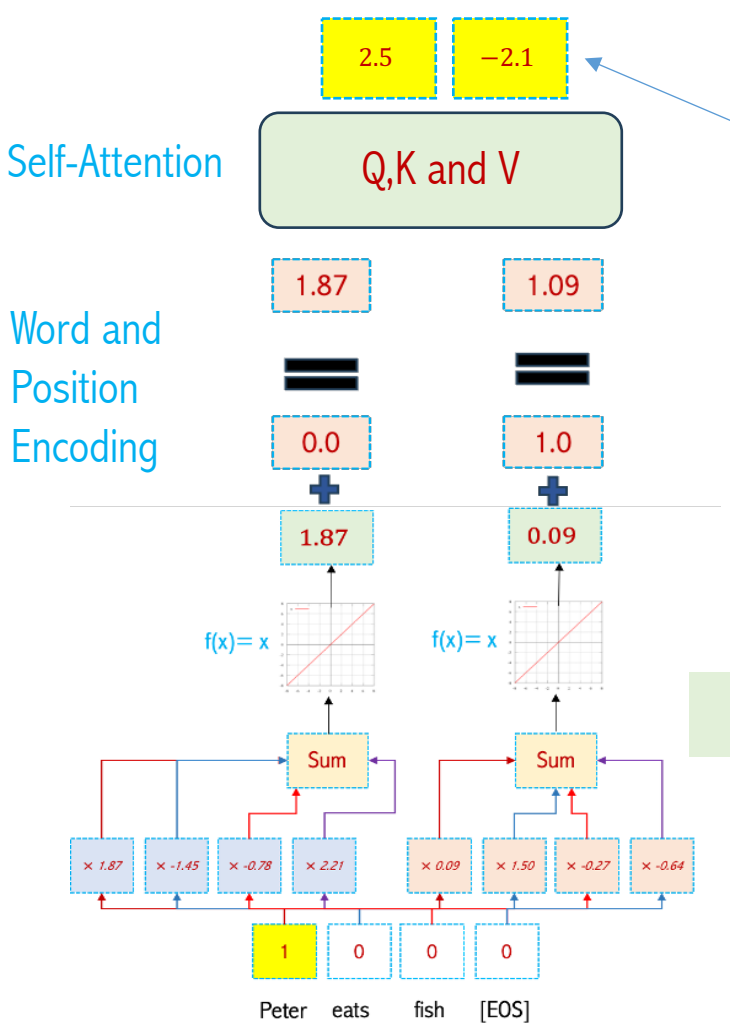


'I' and 'Watch' relative to their similarity to 'I', are the the self-attention values for 'I'

Relationship Among Words: Detail



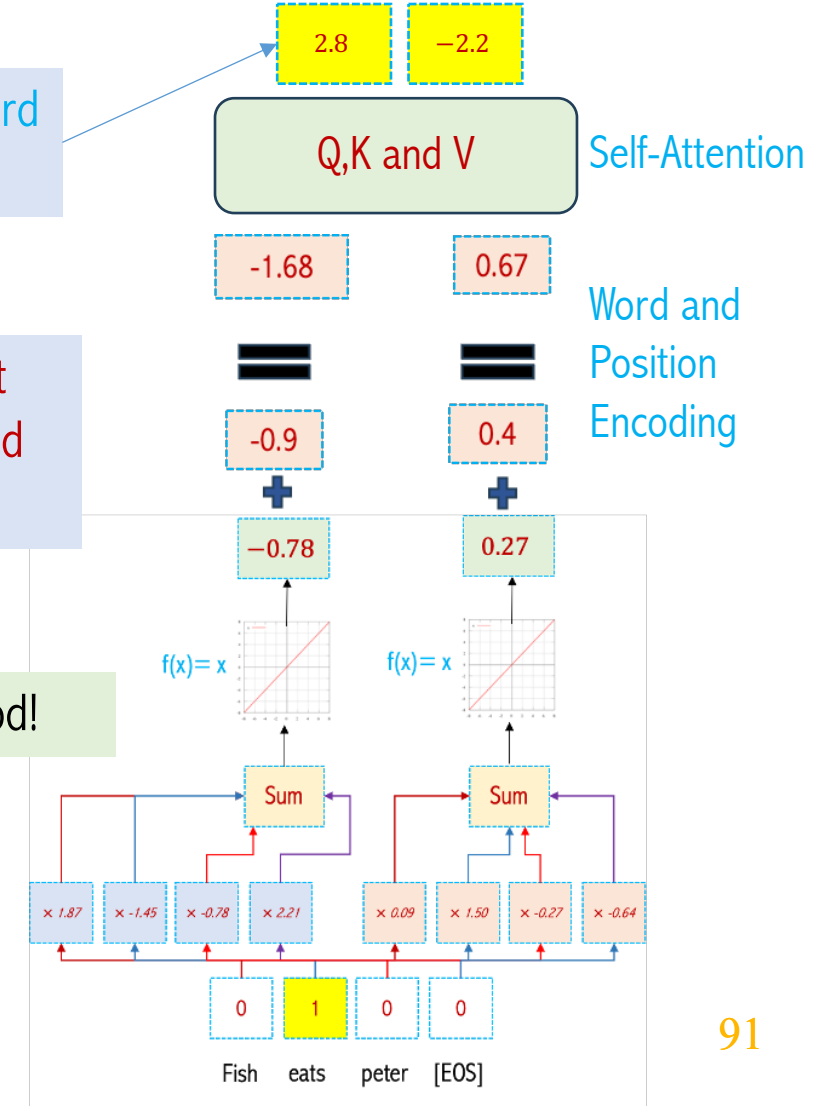
Relationship Among Words: Detail



Pay attention: Self-attention values for each word contain input from all of the other words

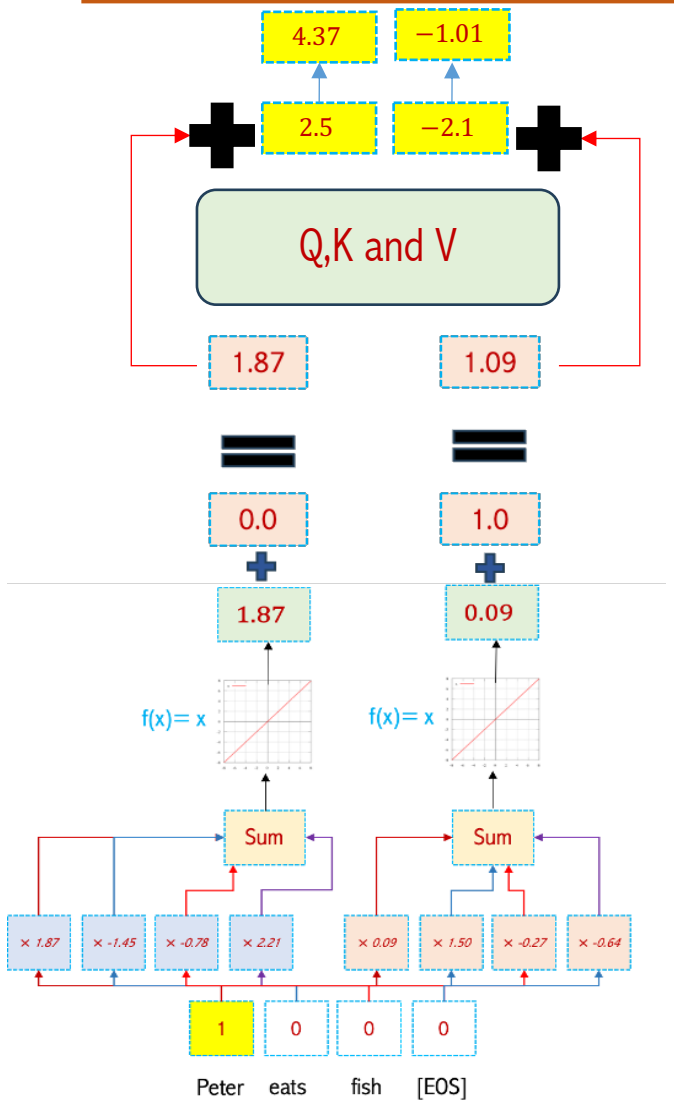
Stack of Self-Attentions to capture different relationship among the words im complicated sentenses and pagraphs

The pizza came out of the oven and it tasted good!



AI VIETNAM
All-in-One Course

Residual Connections

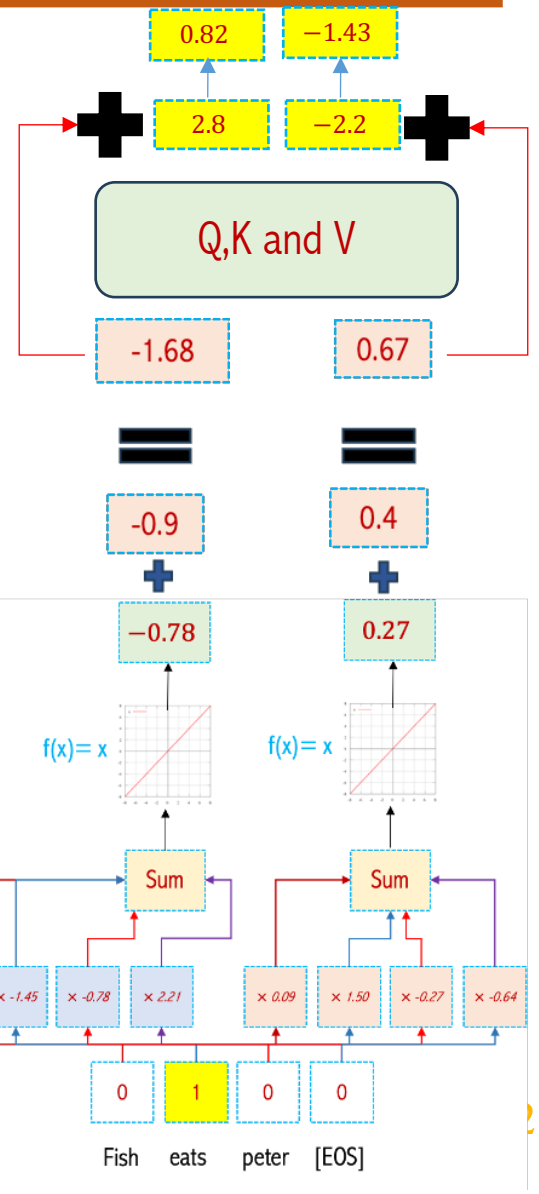


Residual Connection ⟷ Easy to train ⟷ Residual Connection

Self-Attention ⟷ Encode the relationship of words ⟷ Self-Attention

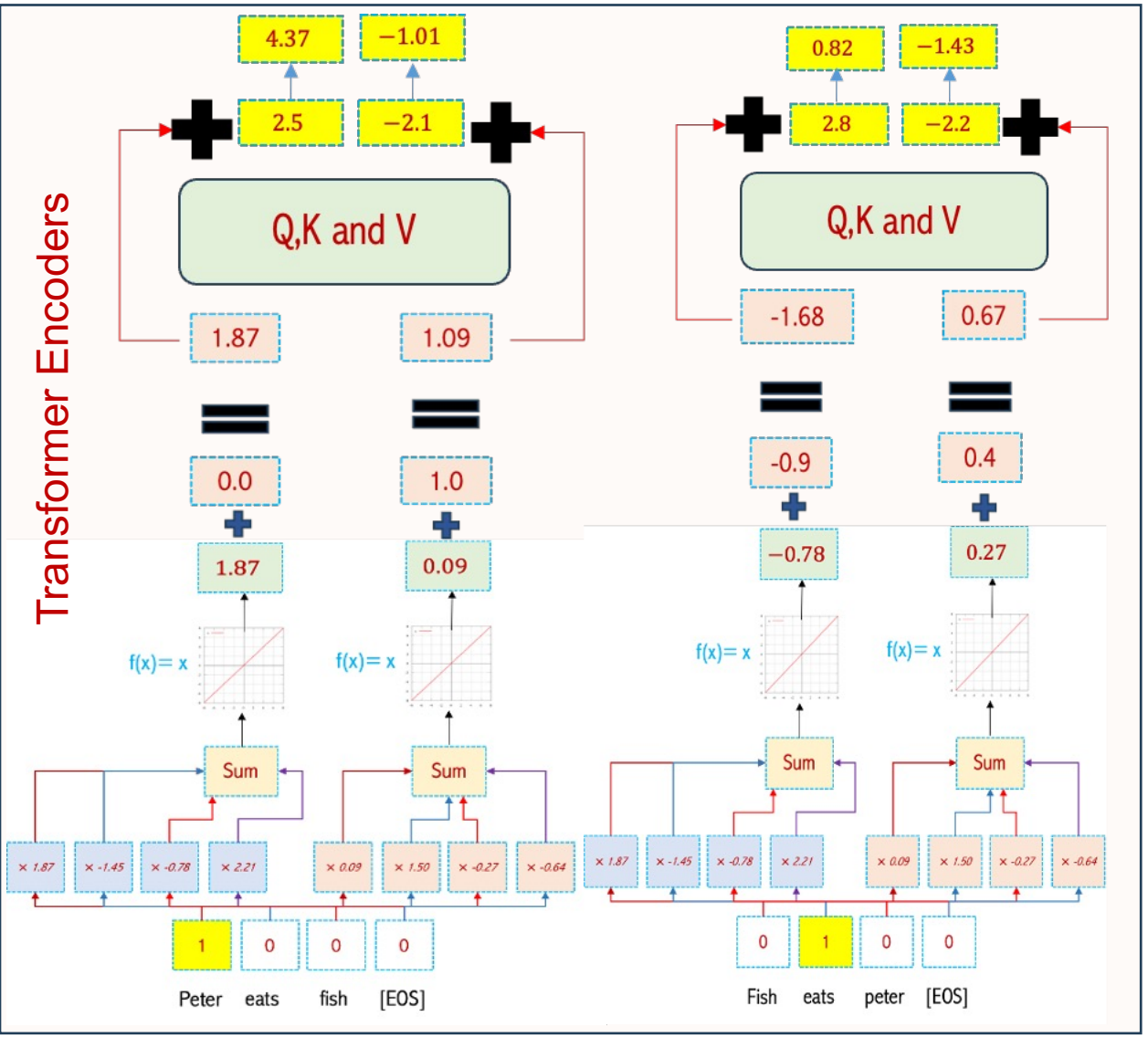
Position Encoding ⟷ Encode the position of the words ⟷ Position Encoding

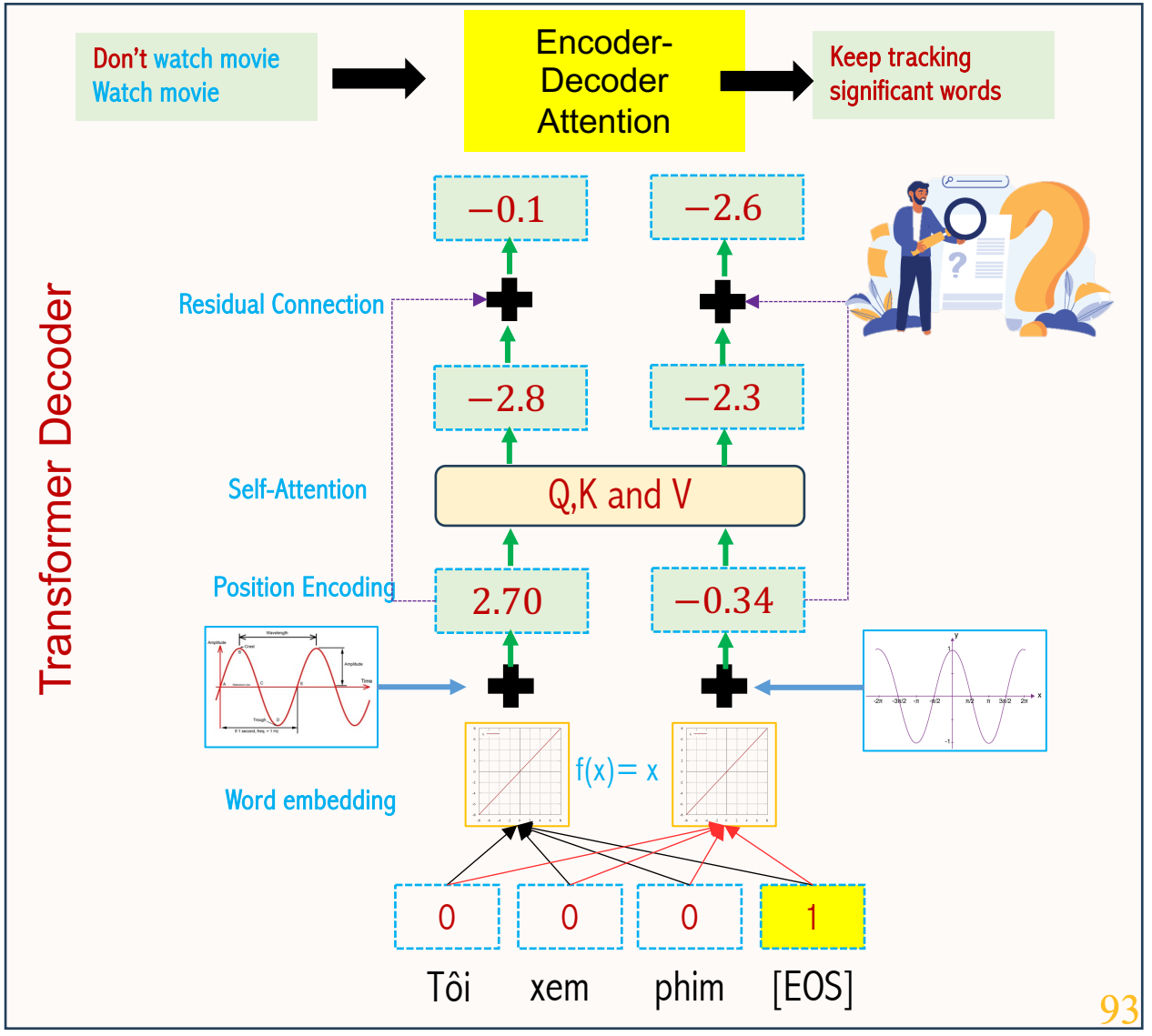
Word embedding ⟷ Encode words into numbers ⟷ Word embedding



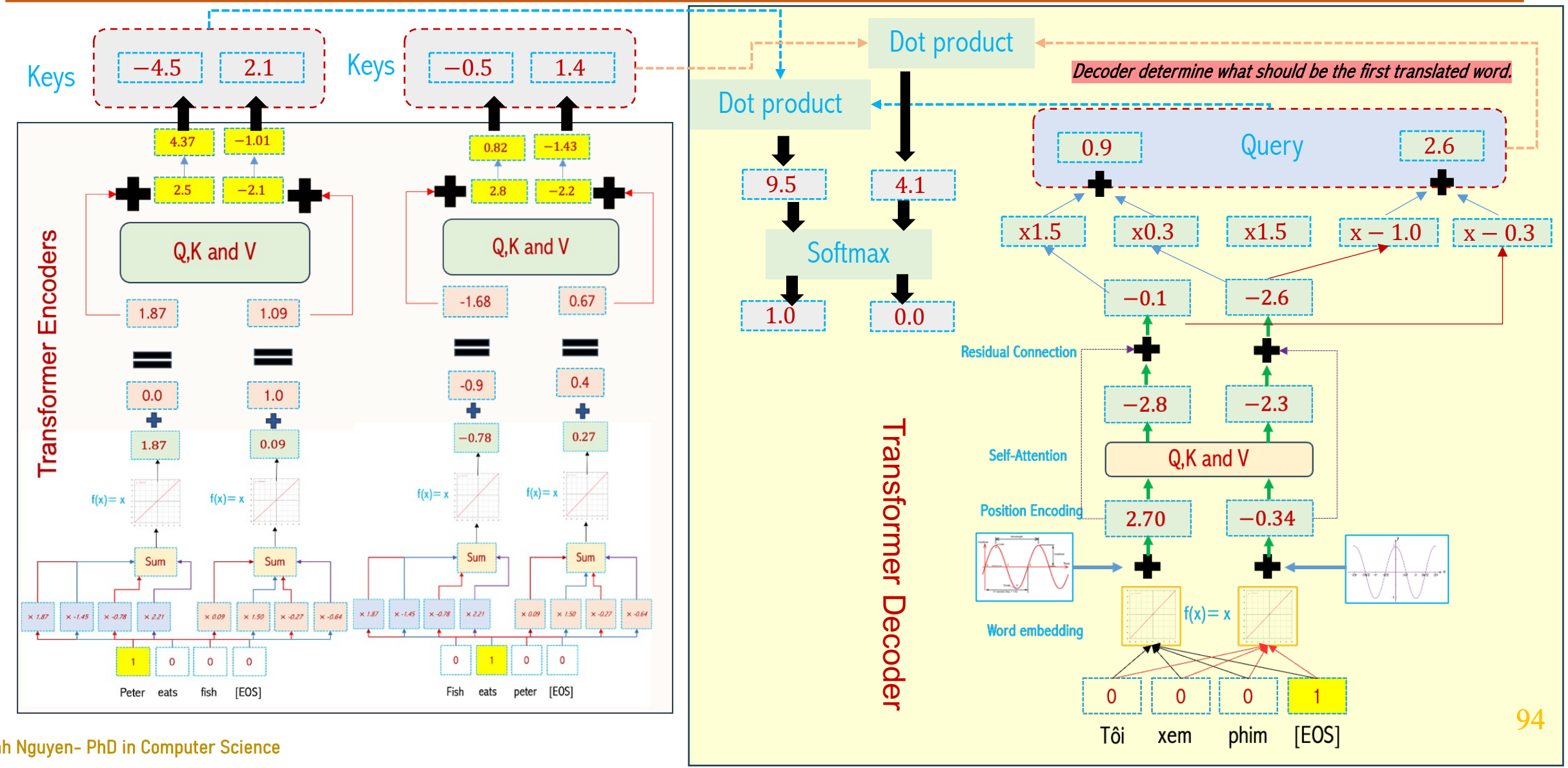
Transformer Encoders:
Encode the input

Transformer Encoder-Decoder

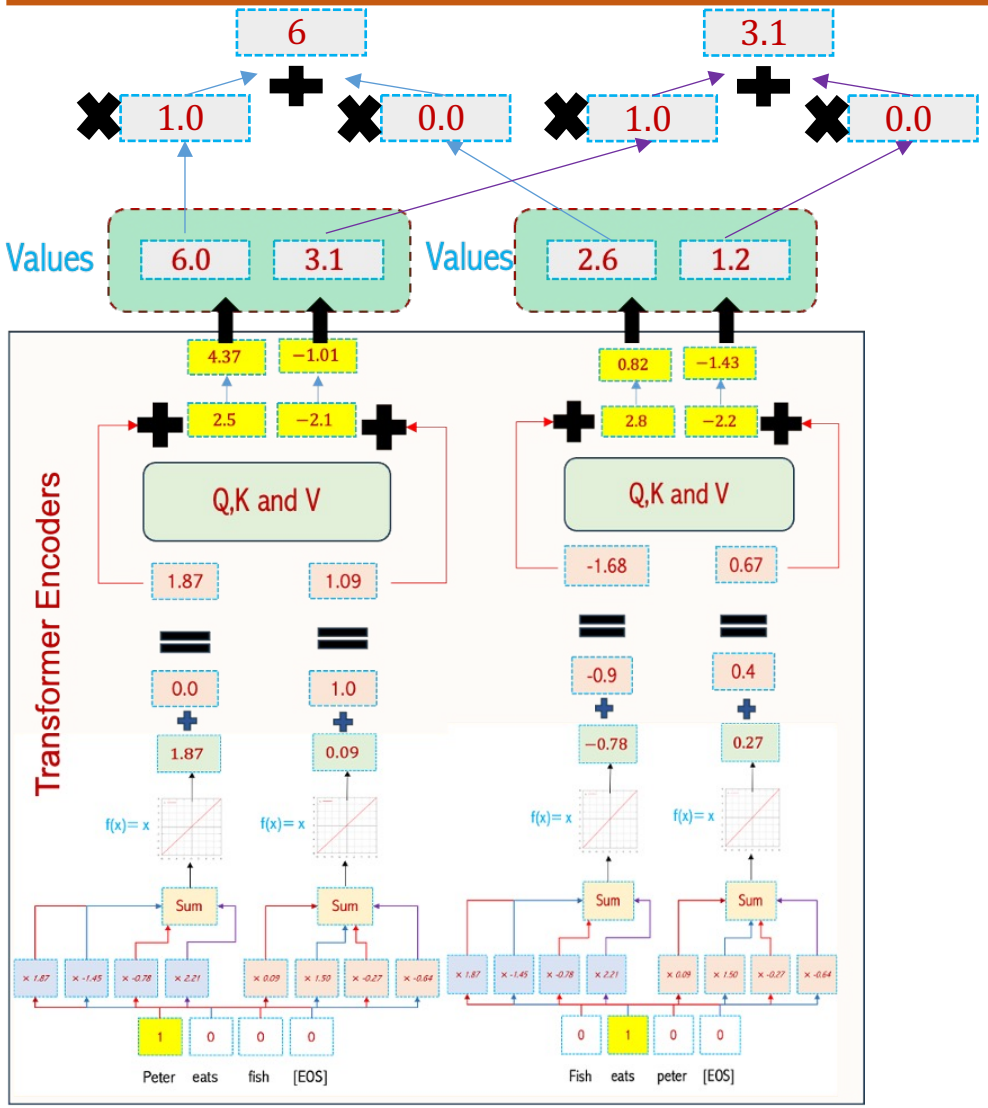


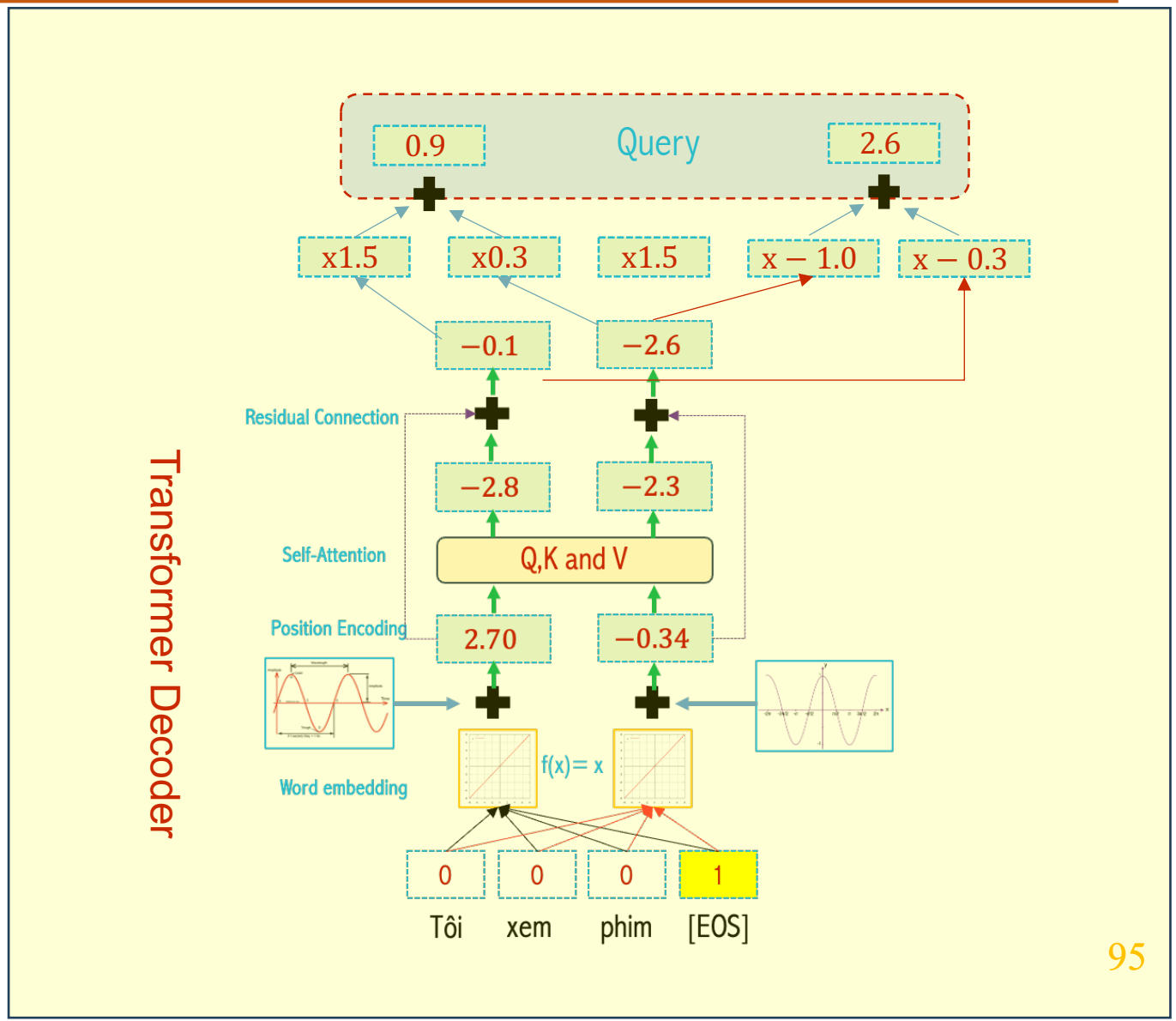


Encoder-Decoder Attention

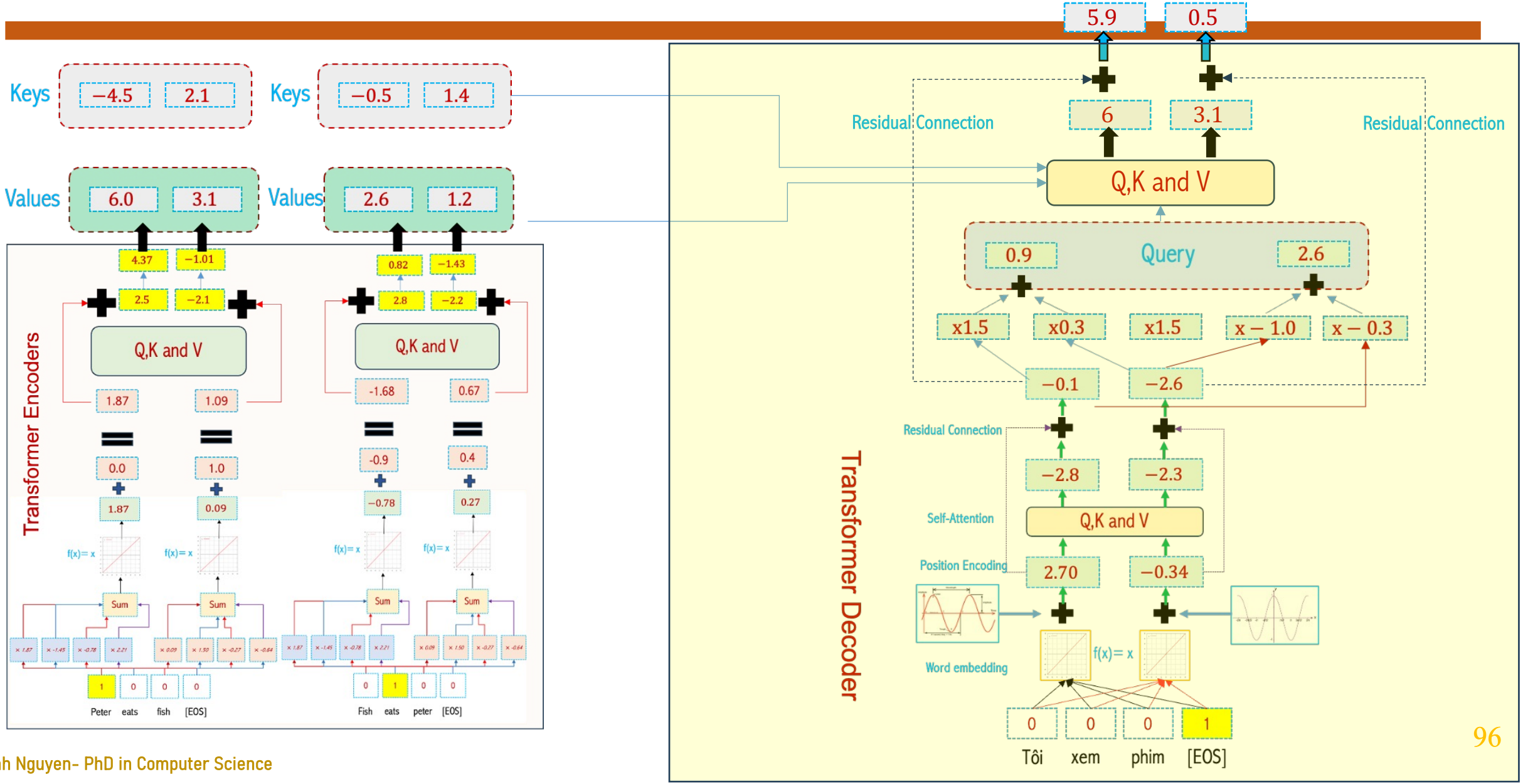


Encoder-Decoder Attention

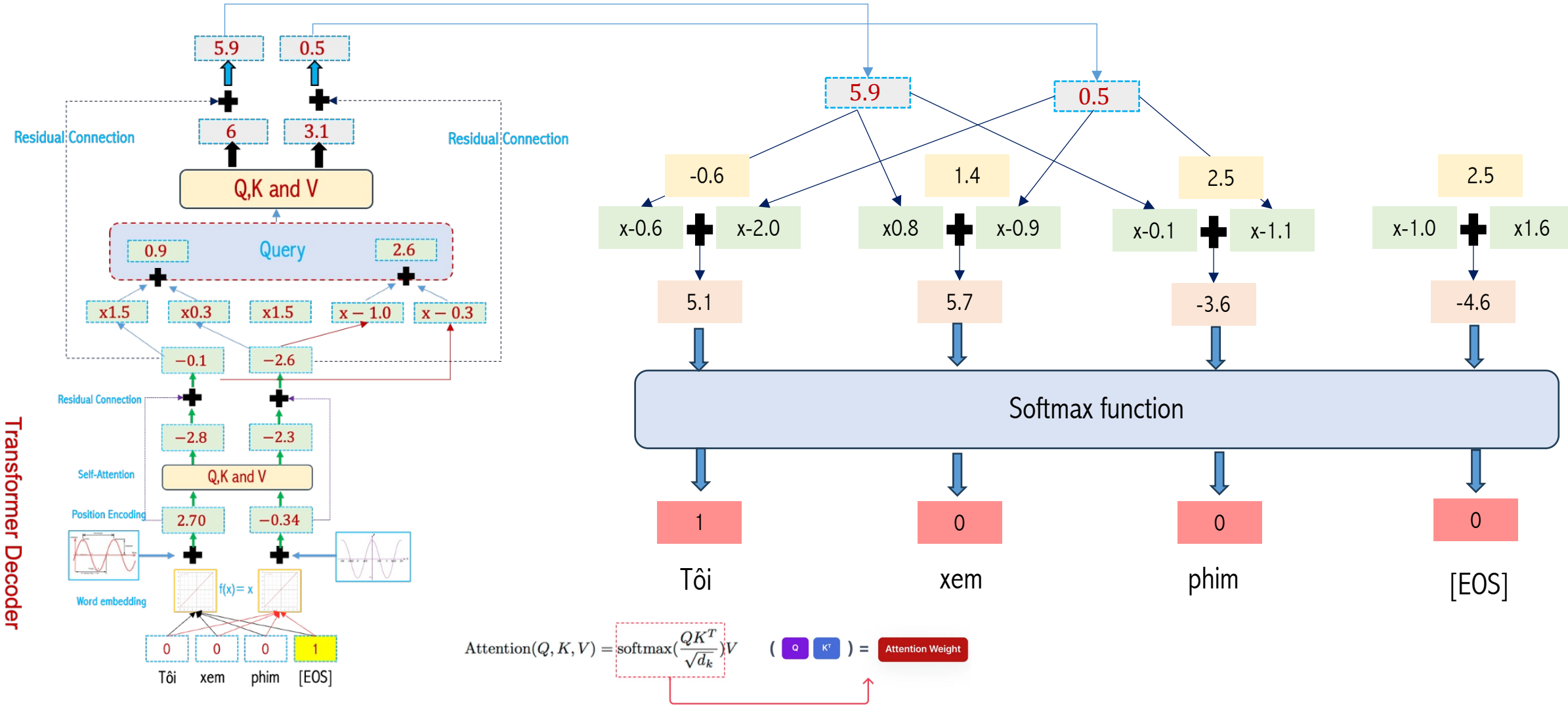




Encoder-Decoder Attention

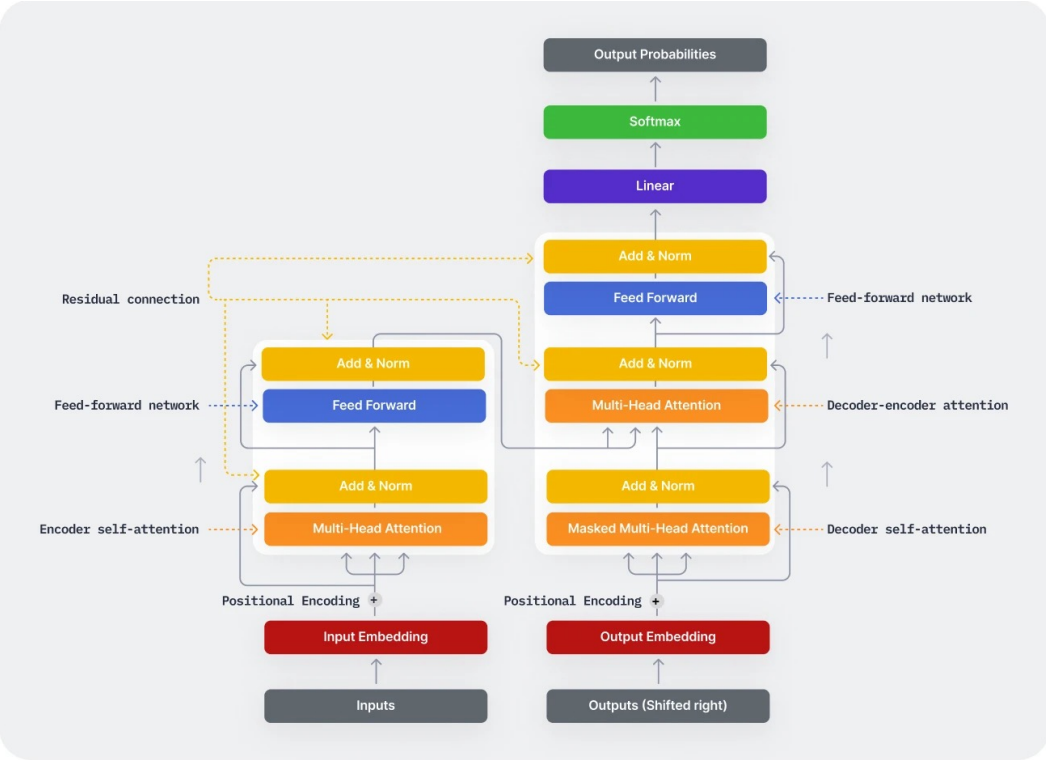


Decoder Output



Translation is correct, but the Decoder does not stop until it outputs an [EOS] token.

Vision Transformer



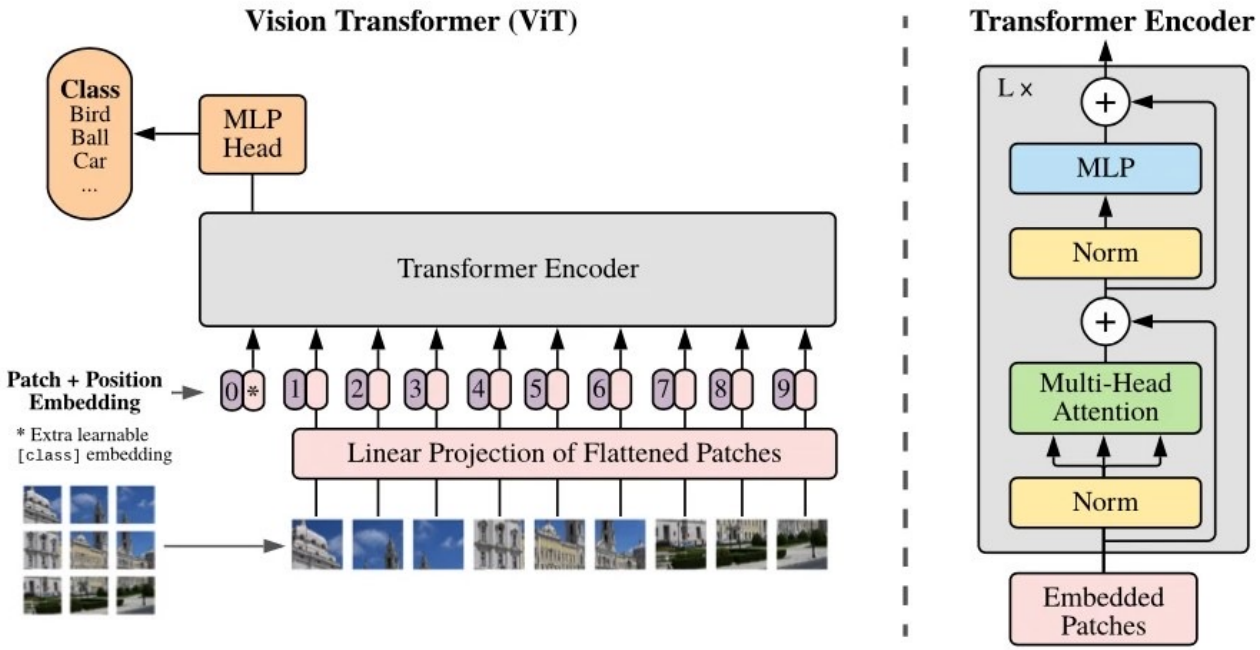
1.Split an image into patches
2.Flatten the patches
3.Produce lower-dimensional linear embeddings from the flattened patches
4.Add positional embeddings
5.Feed the sequence as an input to a standard transformer encoder
6.Pretrain the model with image labels (fully supervised on a huge dataset)
7.Finetune on the downstream dataset for image classification

An Image is Worth 16*16 Words: Transformers for Image Recognition at Scale," published at ICLR 2021.

The ViT model represents an input image as a series of image patches, like the series of word embeddings used when using transformers to text, and directly predicts class labels for the image.

Vision Transformer

1.Split an image into patches
2.Flatten the patches
3.Produce lower-dimensional linear embeddings from the flattened patches
4.Add positional embeddings
5.Feed the sequence as an input to a standard transformer encoder
6.Pretrain the model with image labels (fully supervised on a huge dataset)
7.Finetune on the downstream dataset for image classification



An Image is Worth 16*16 Words: Transformers for Image Recognition at Scale," published at ICLR 2021.

The ViT model represents an input image as a series of image patches, like the series of word embeddings used when using transformers to text, and directly predicts class labels for the image.


Thank
you