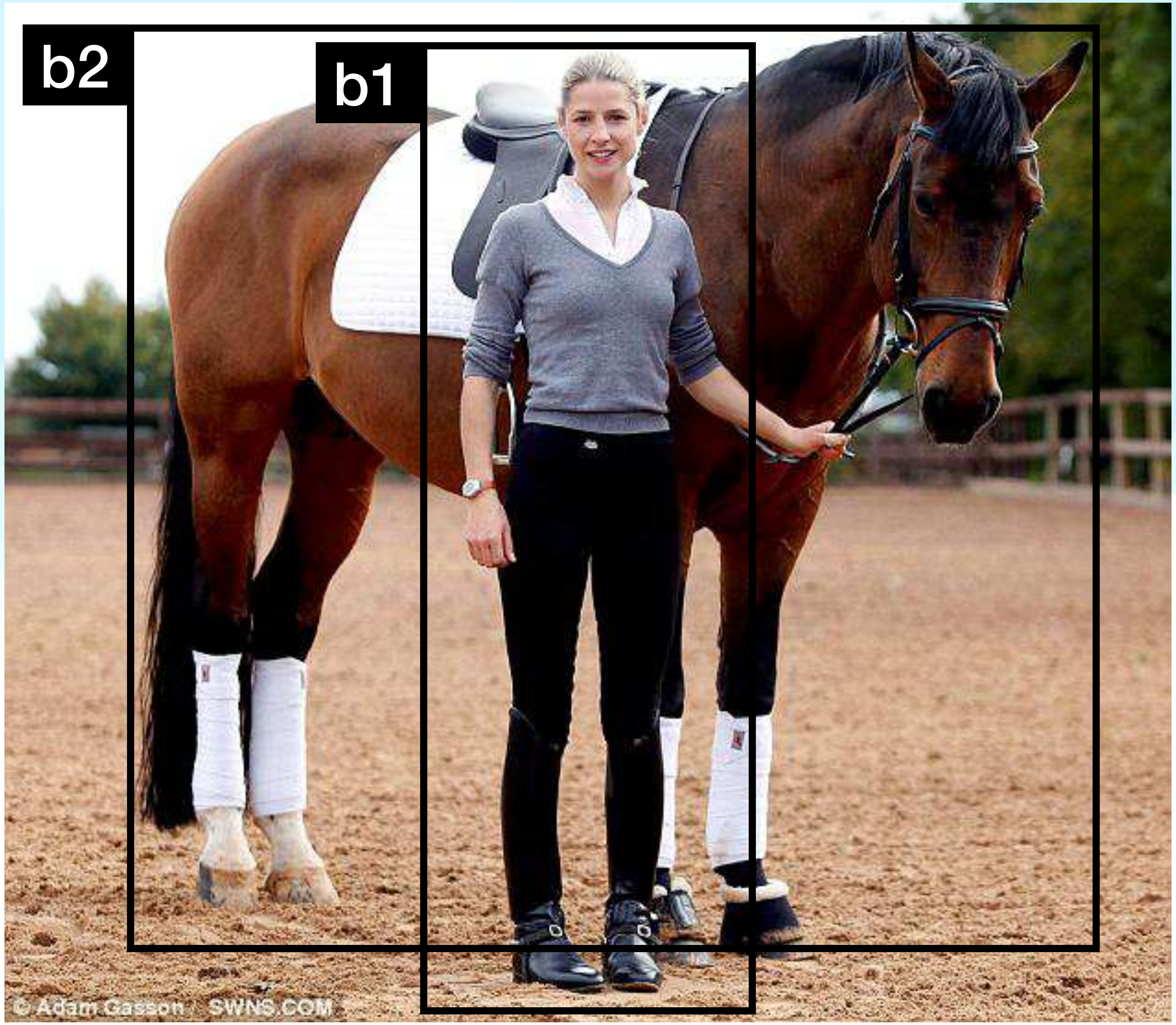

YOLO-V1
Let's Dig Deeper
randomly selection
image dataset
448
label encoding
loss computation
YOLO v1 Con. Network
30
7
predicted map
448

Object Detection

- Input:
  - Image
- Output:
  - {b1,b2,…bn} bounding boxes of n detected objects
  - {c1,c2,…cn} class labels of all detected objects

Object Detection

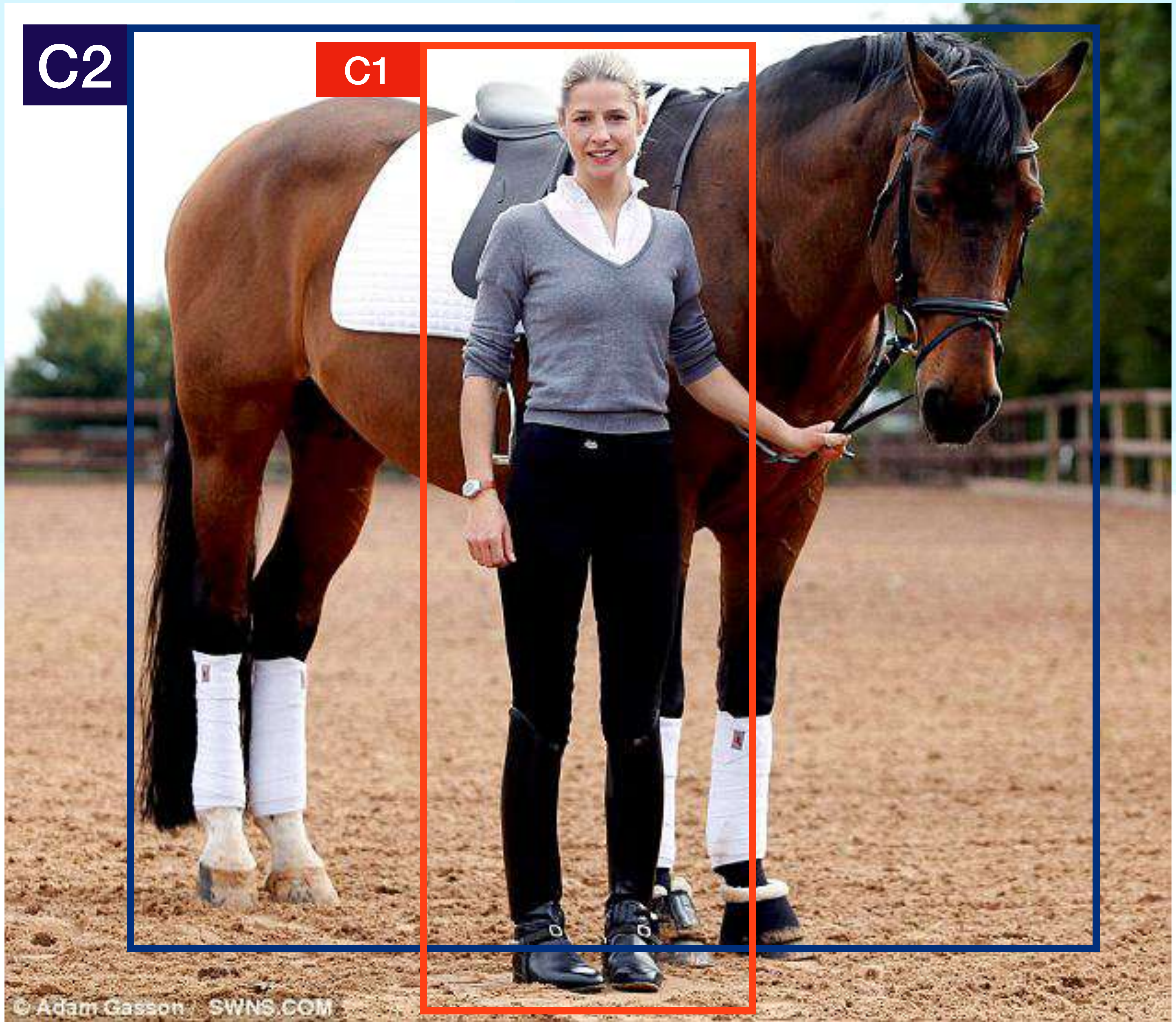
- Input:
  - Image
- Output:
  - {b1,b2,…bn} bounding boxes of n detected objects
  - {c1,c2,…cn} class labels of all detected objects



- Input:
  - Image
- Output:
  - {b1,b2,…bn} bounding boxes of n detected objects
  - {c1,c2,…cn} class labels of all detected objects

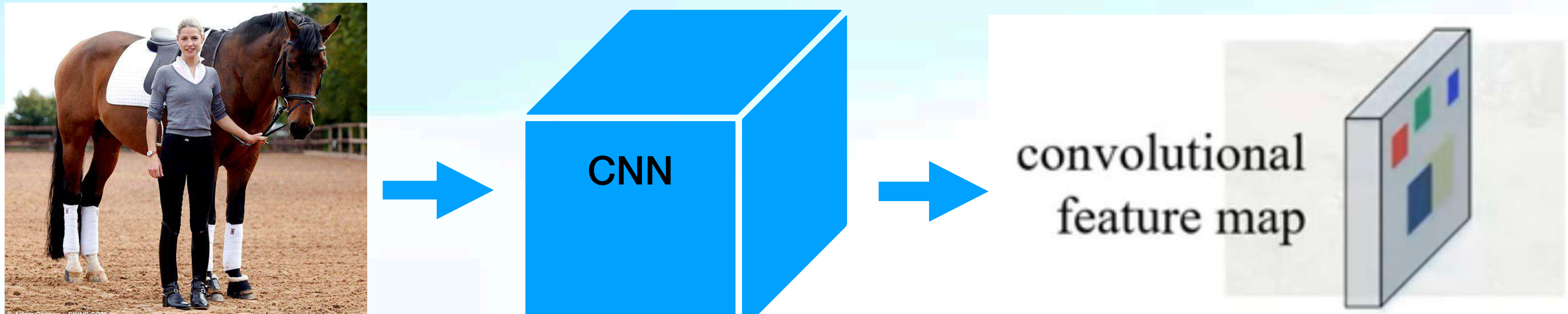
C1 = Person

C2 = Horse



Object Detection

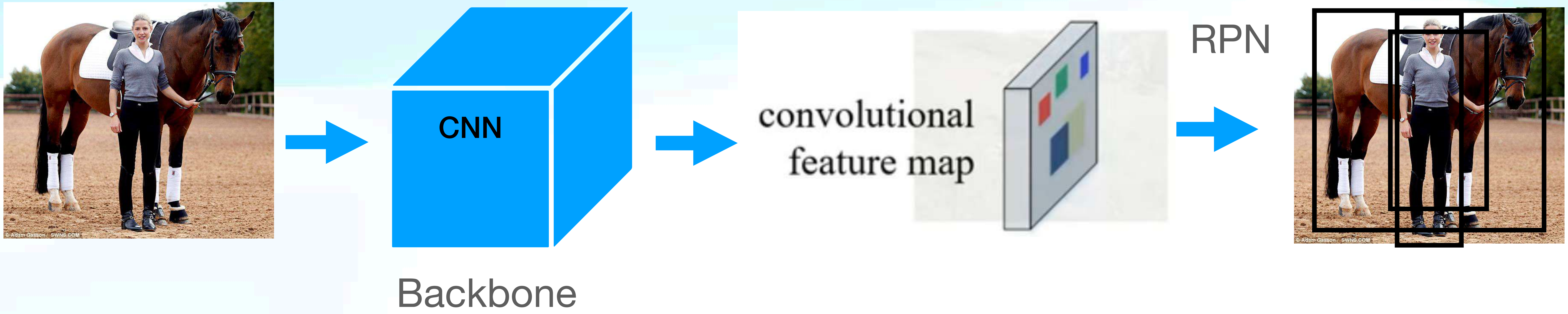
- Most algorithms repurpose classifiers to perform object detection.
- Faster-RCNN: Uses RPN to predict boxes and classifier to predict class probabilities.



Backbone

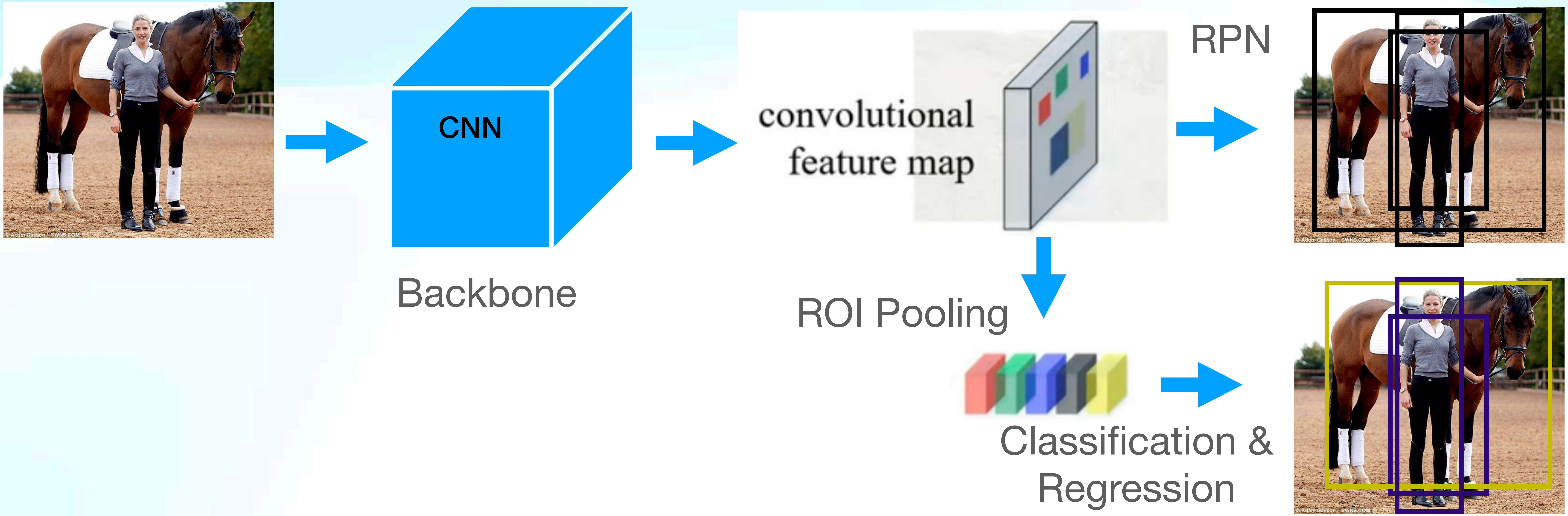
Object Detection

- Most algorithms repurpose classifiers to perform object detection.
- Faster-RCNN: Uses RPN to predict boxes and classifier to predict class probabilities.



Object Detection

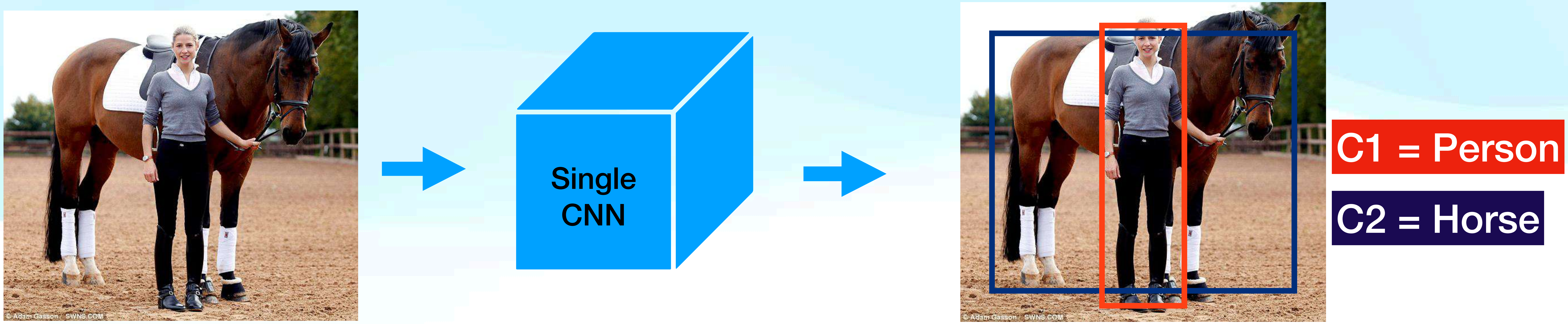
- Most algorithms repurpose classifiers to perform object detection.
- Faster-RCNN: Uses RPN to predict boxes and classifier to predict class probabilities.



Drawbacks

- Multi-stage pipelines
- Each component trained separately
- Complex and not useful for real-time applications
- Not generalisable to other domains

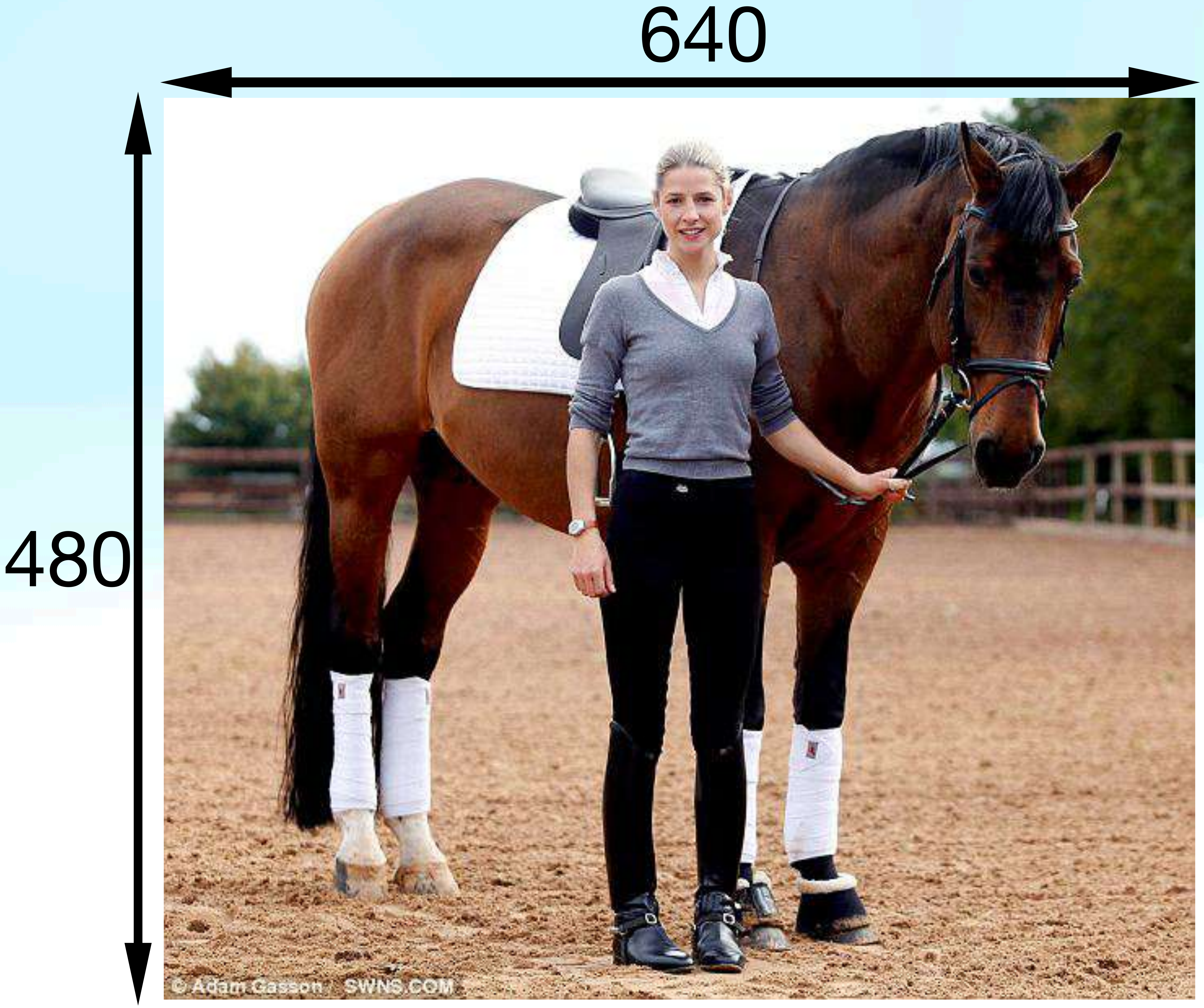
Idea of YOLO



Reframe object detection as a single stage regression problem

Steps in YOLO

- Take input image

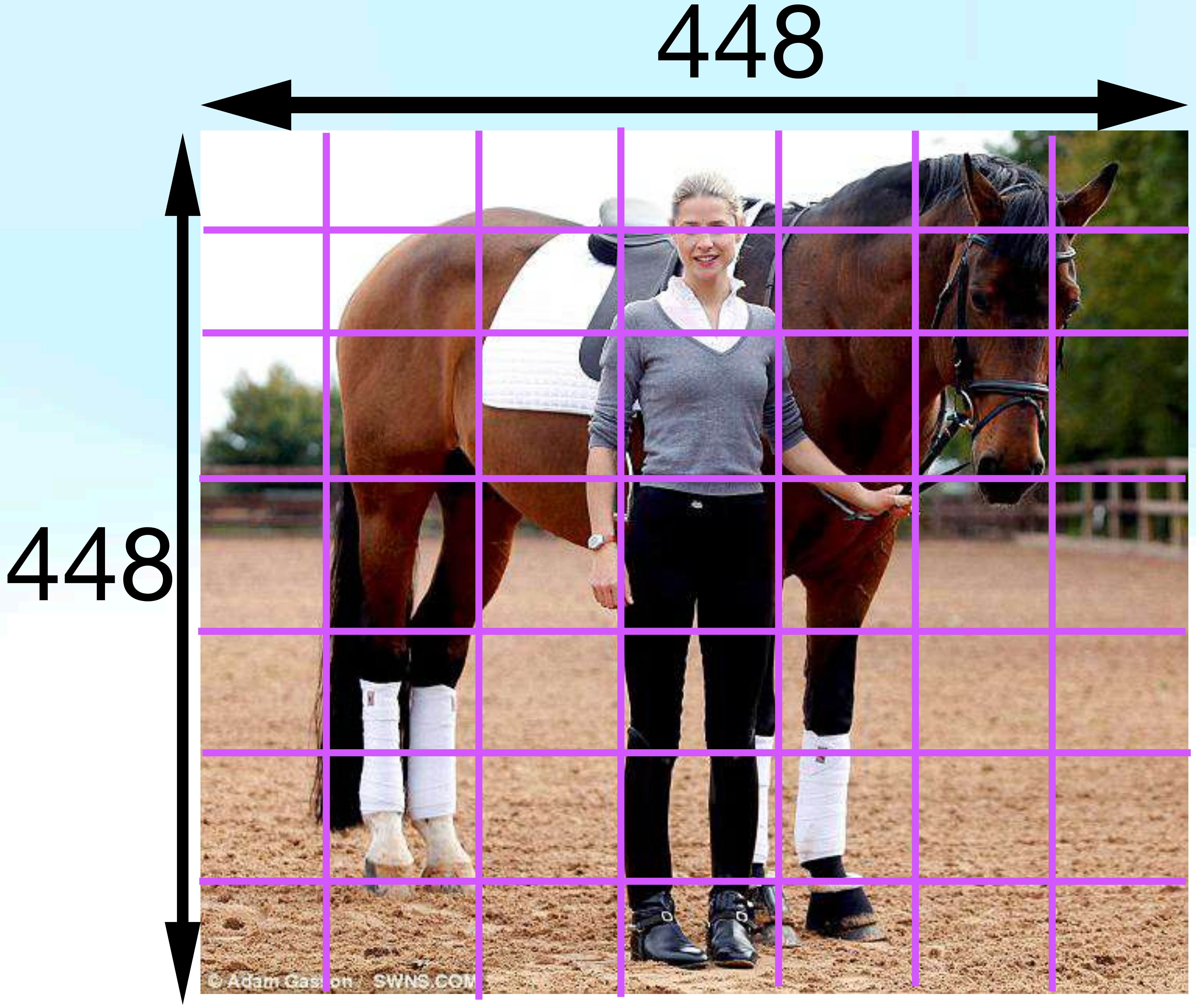


Steps in YOLO

- Take input image
- Resize to 448x448

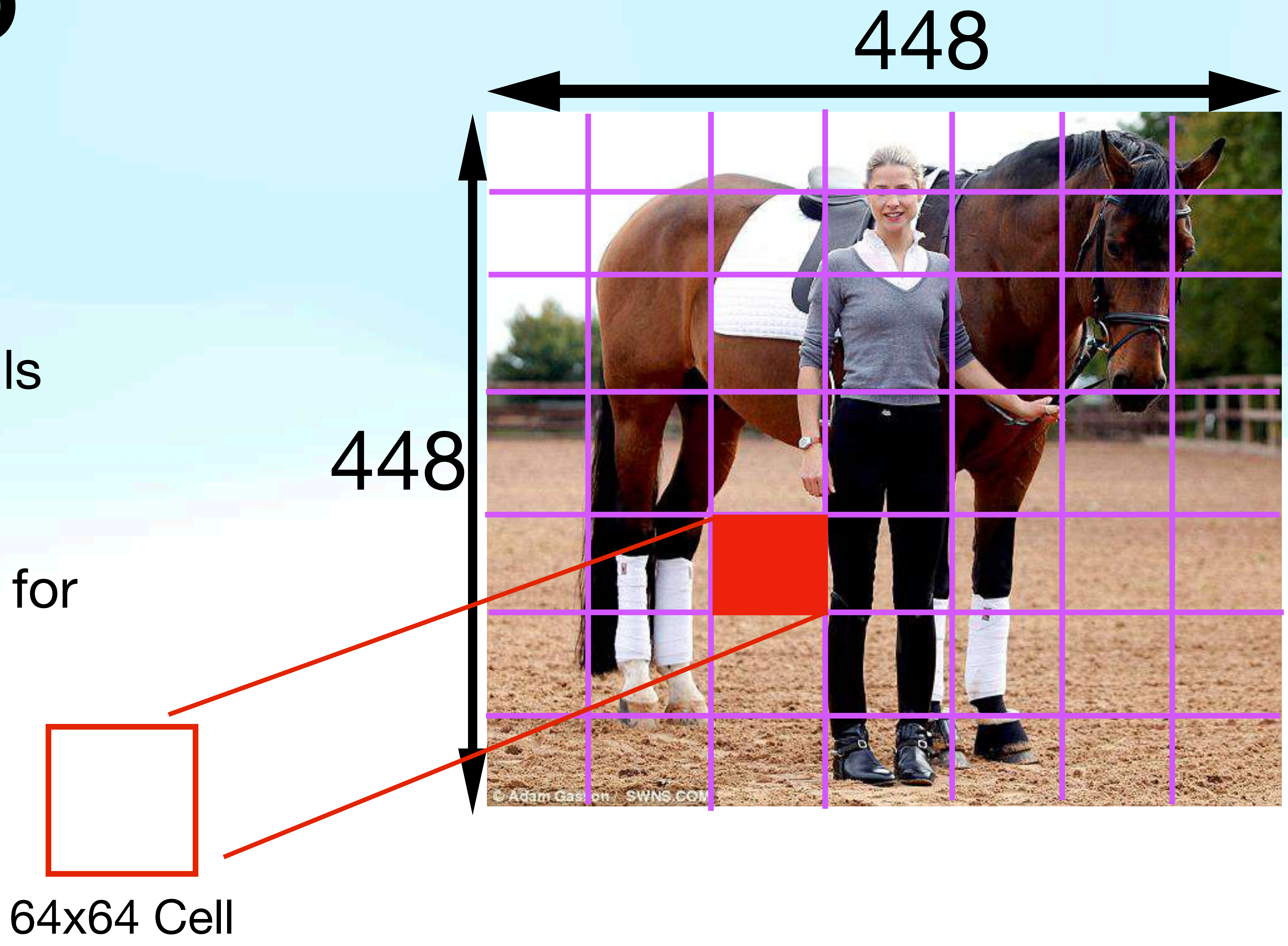
Steps in YOLO

- Take input image
- Resize to 448x448
- Divide into SxS Grid cells
- S=7 in paper



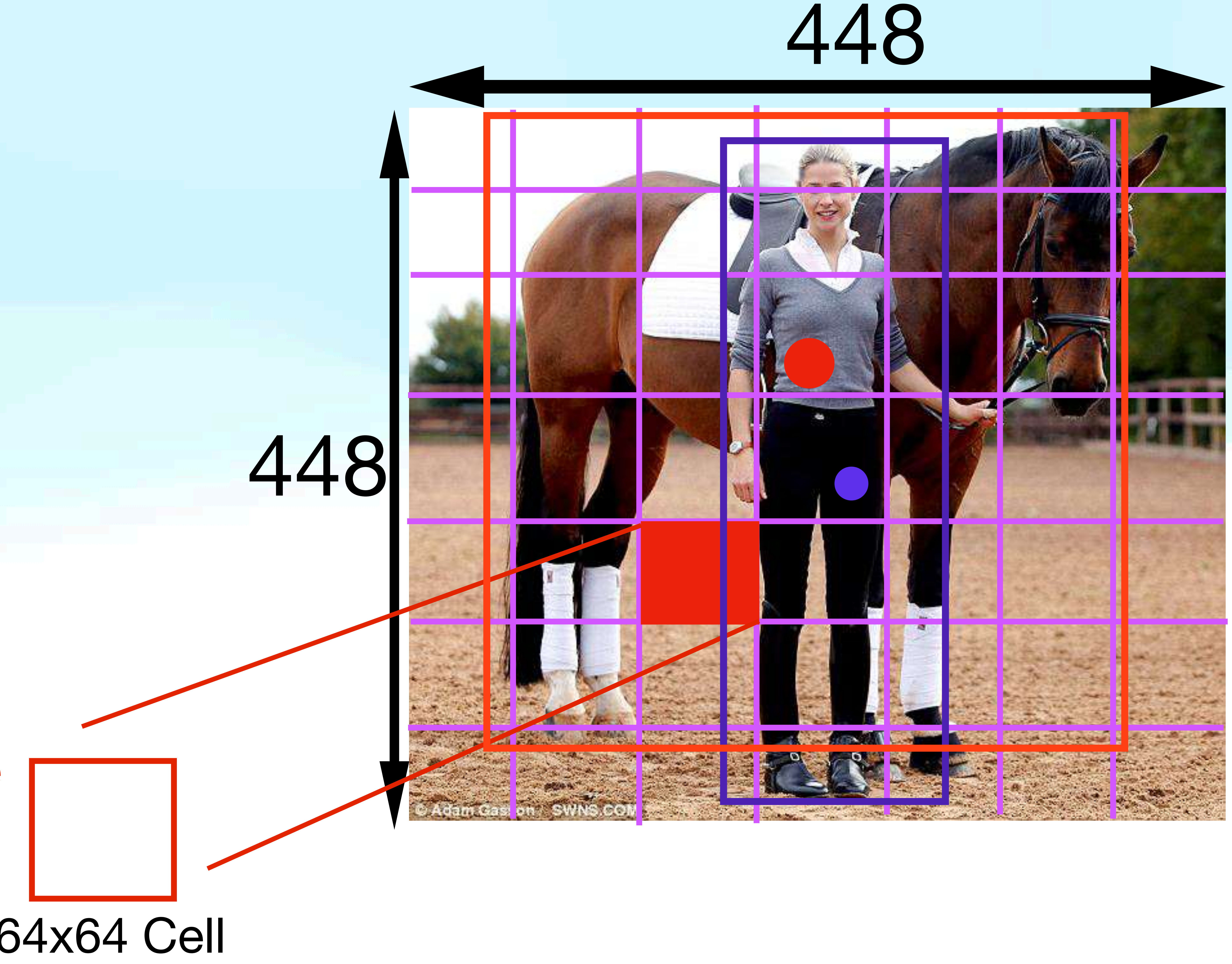
Steps in YOLO

- Take input image
- Resize to 448x448
- Divide into SxS Grid cells
- S=7 in paper
- Each cell is responsible for predicting one object



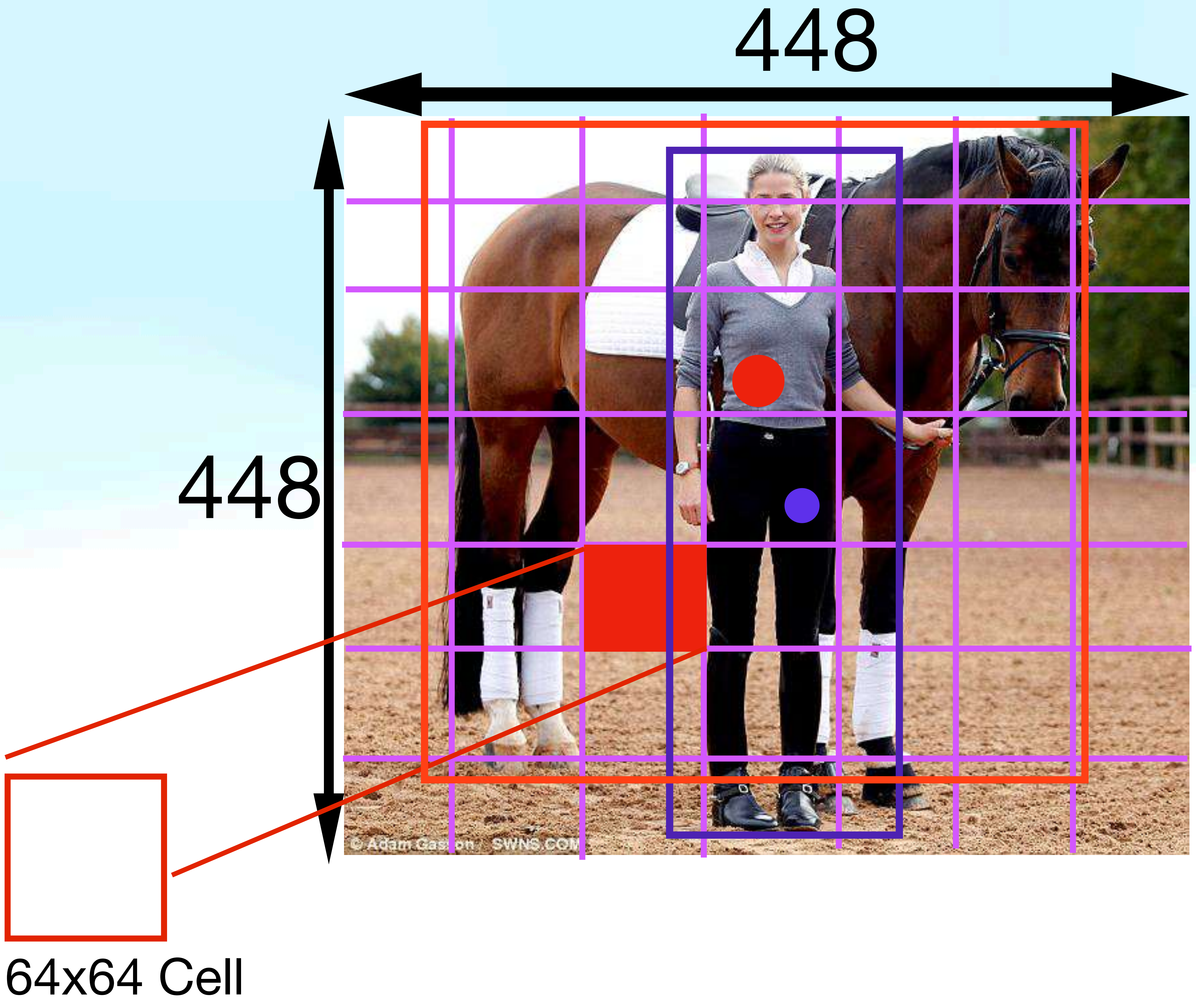
Steps in YOLO

- Take input image
- Resize to 448x448
- Divide into SxS Grid cells
- S=7 in paper
- Each cell is responsible for predicting one object
- Which cells are responsible for person and horse?



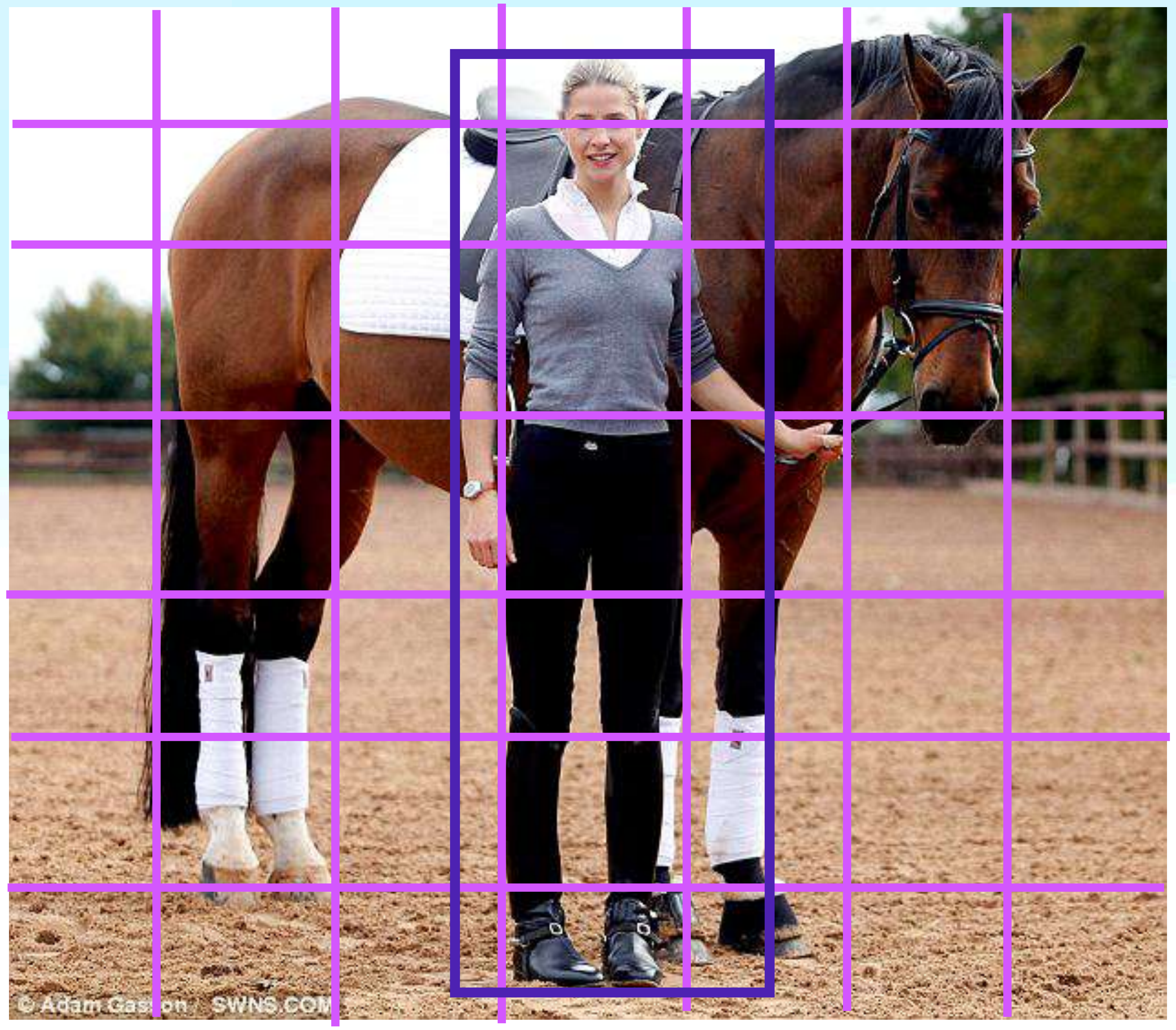
Steps in YOLO

- Take input image
- Resize to 448x448
- Divide into SxS Grid cells
- S=7 in paper
- Each cell is responsible for predicting one object
- Which cell is responsible?
- Where Center of object falls into



Bounding boxes

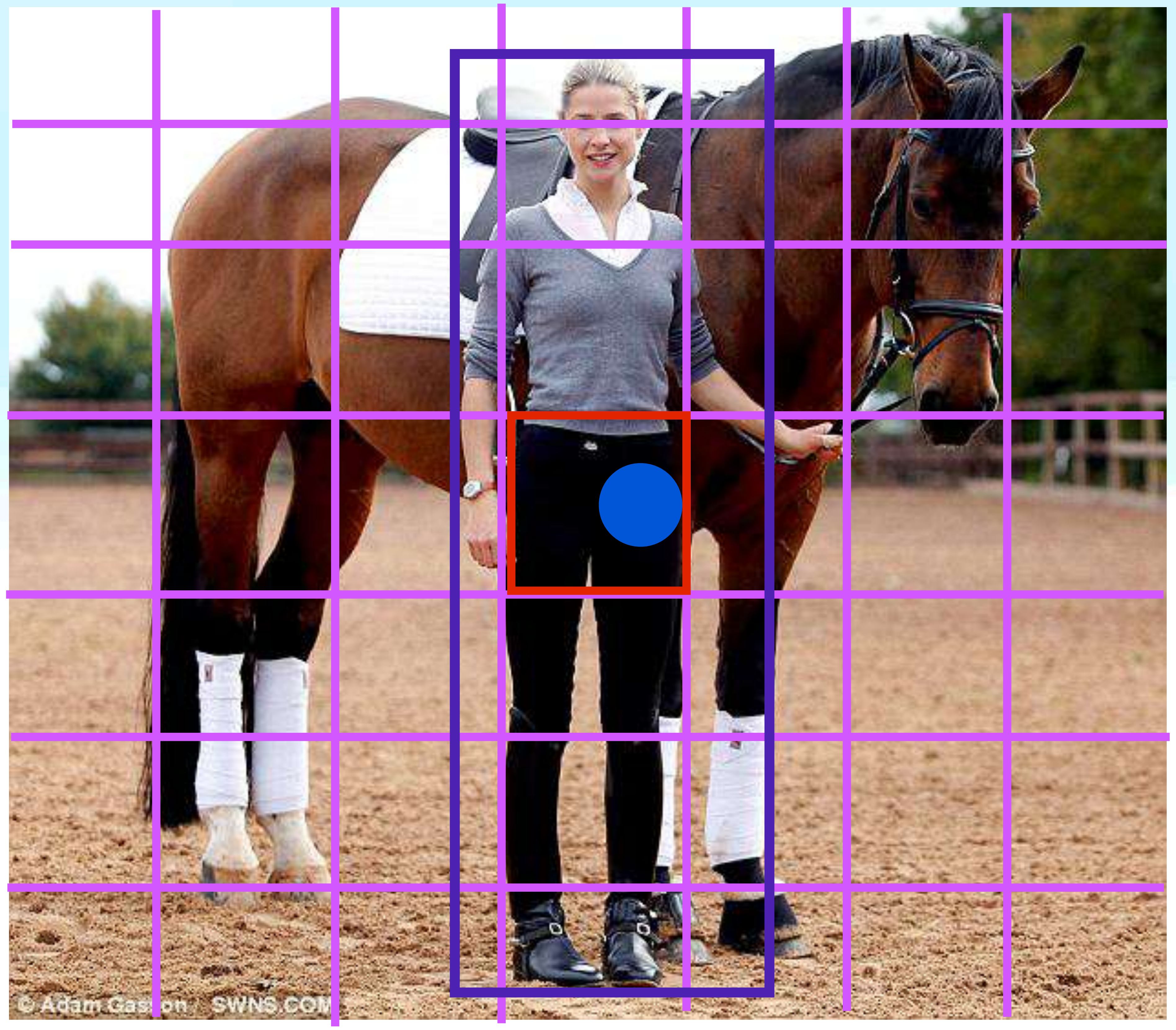
- Box - x,y,w,h



Bounding boxes

- Box - x,y,w,h
- How are these encoded?

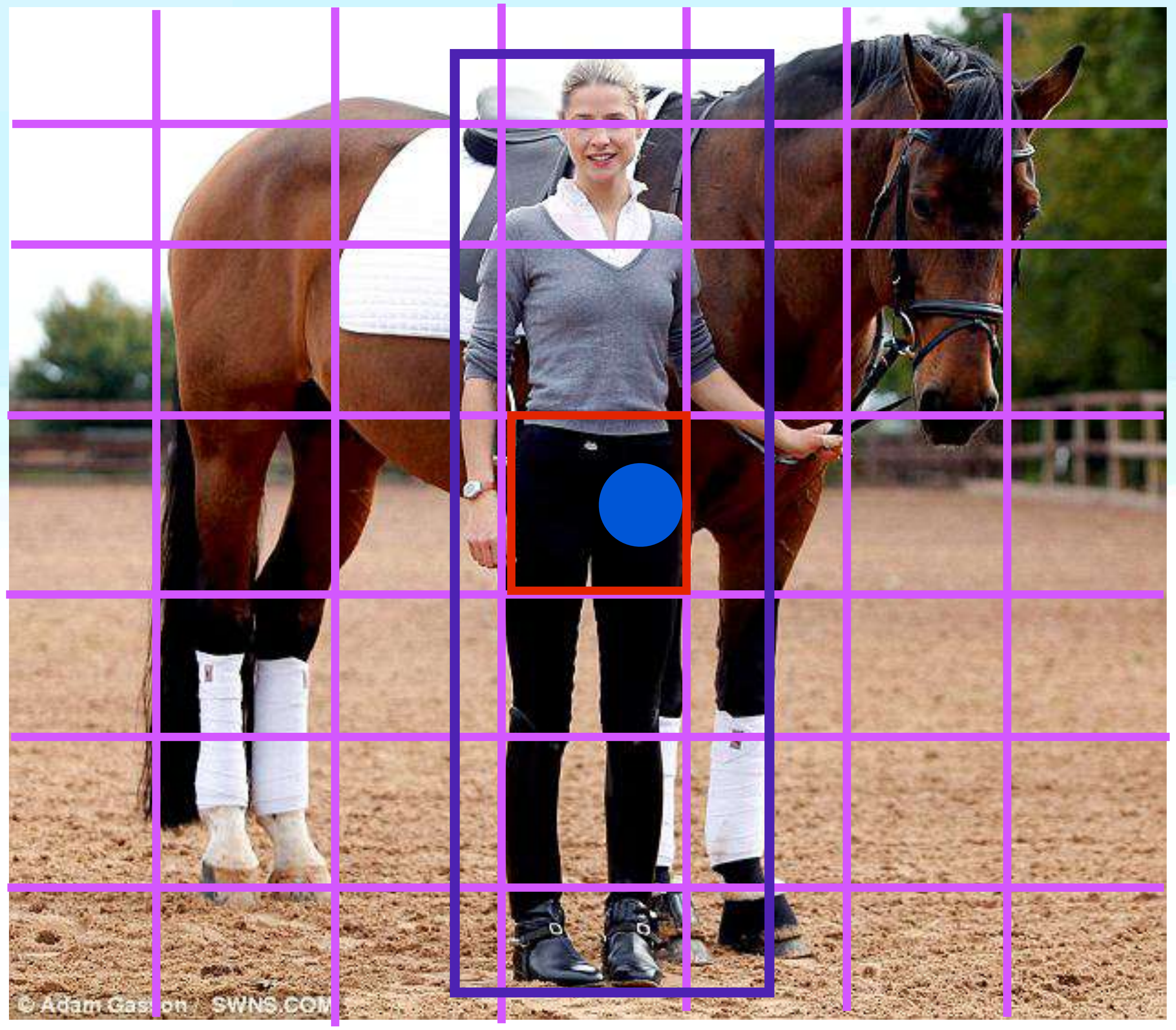
(200, 311, 142, 250)



Bounding boxes

- Box - x,y,w,h
- How are these encoded?
- Relative to Grid cell that the object Center falls into.

$(200, 311, 142, 250)$



Bounding boxes

- Center point (x,y): Relative to anchor that (x,y) falls into.

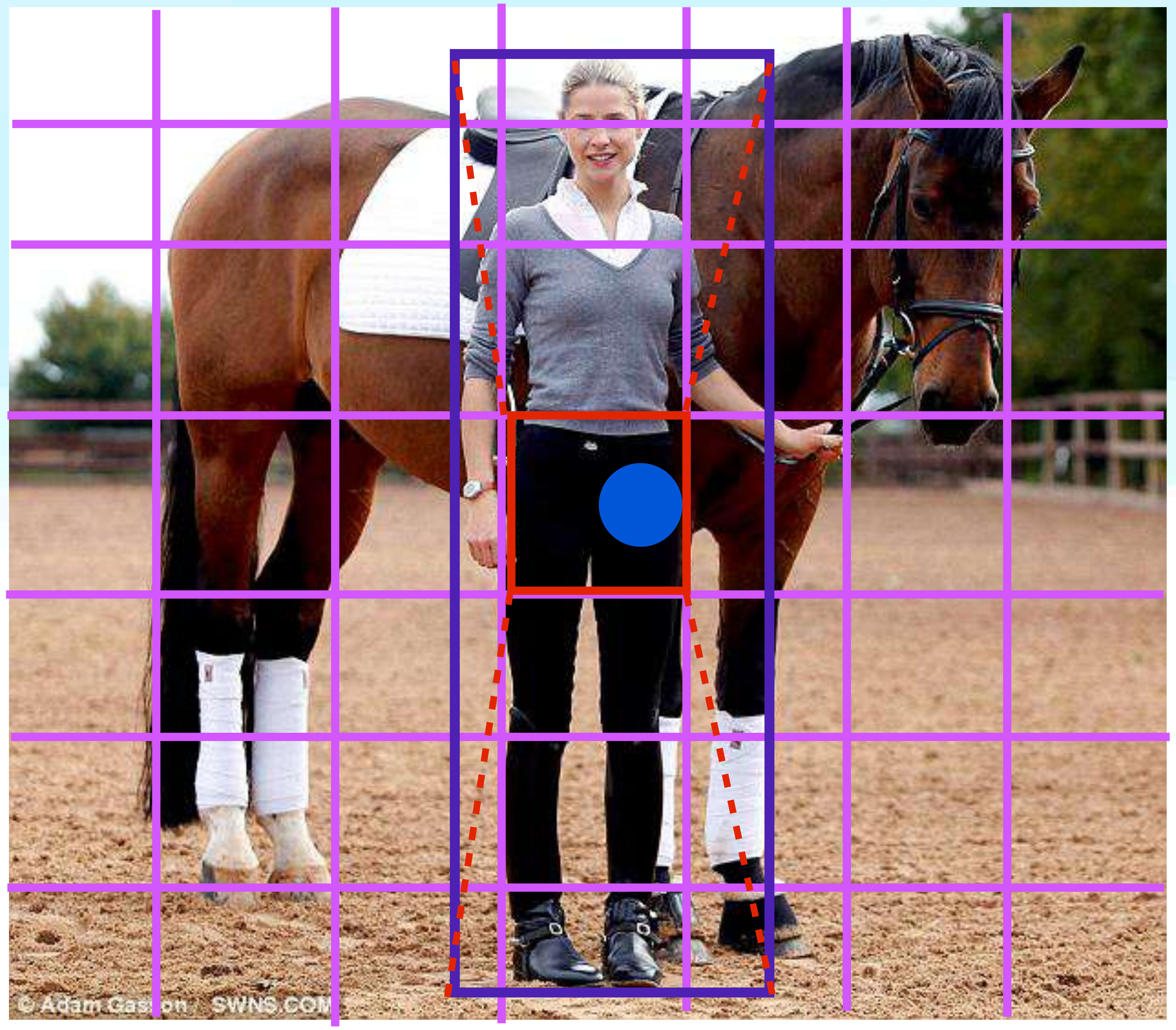
$$\Delta x = (x - x_a)/64$$
$$\Delta y = (y - y_a)/64$$

(x_a, y_a): the coordinate of left-top point

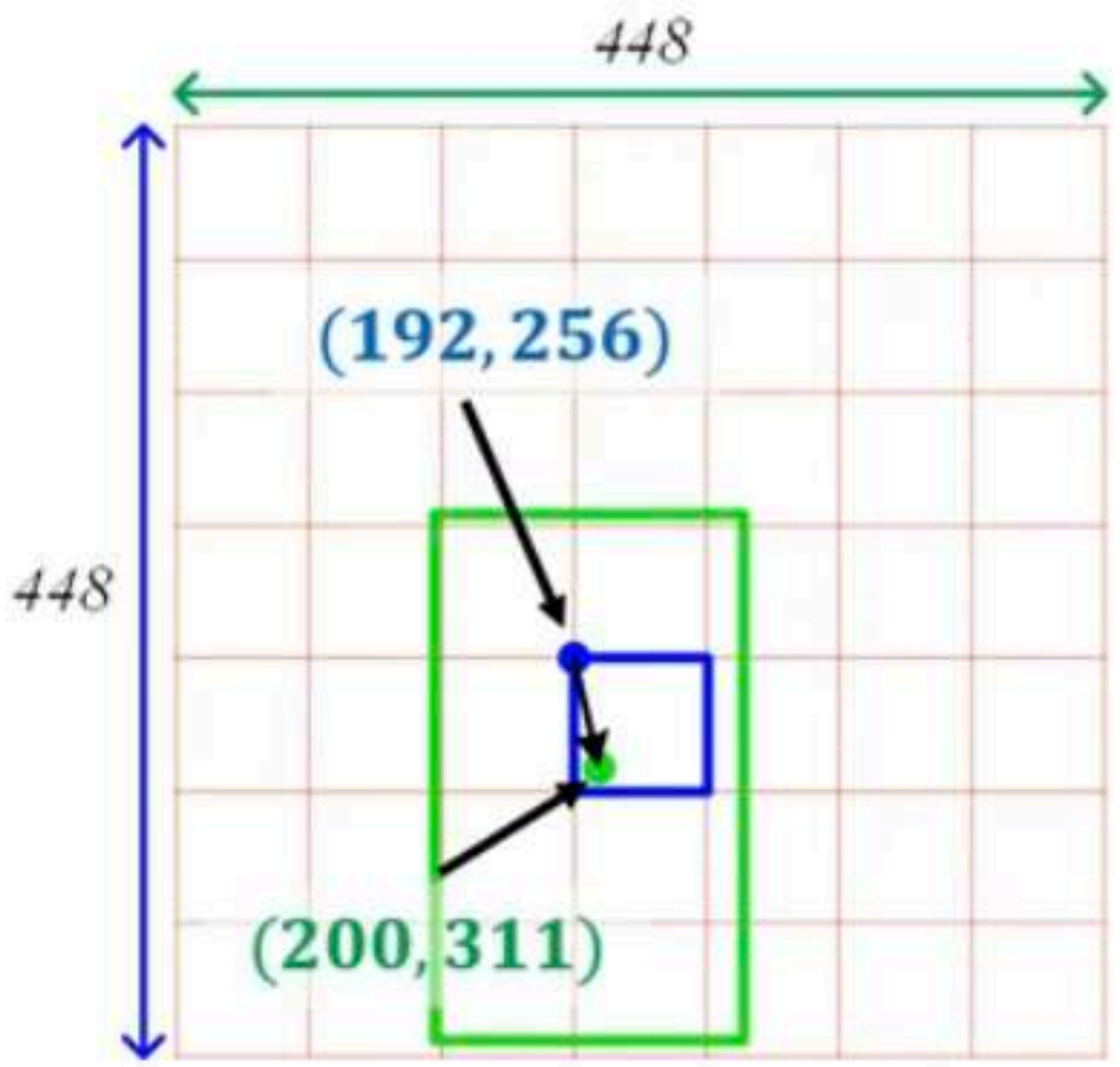
- Width/height (w,h): relative to the whole image

$$\Delta w = w/448$$
$$\Delta h = h/448$$

$(200, 311, 142, 250)$



Example Calculation - Targets



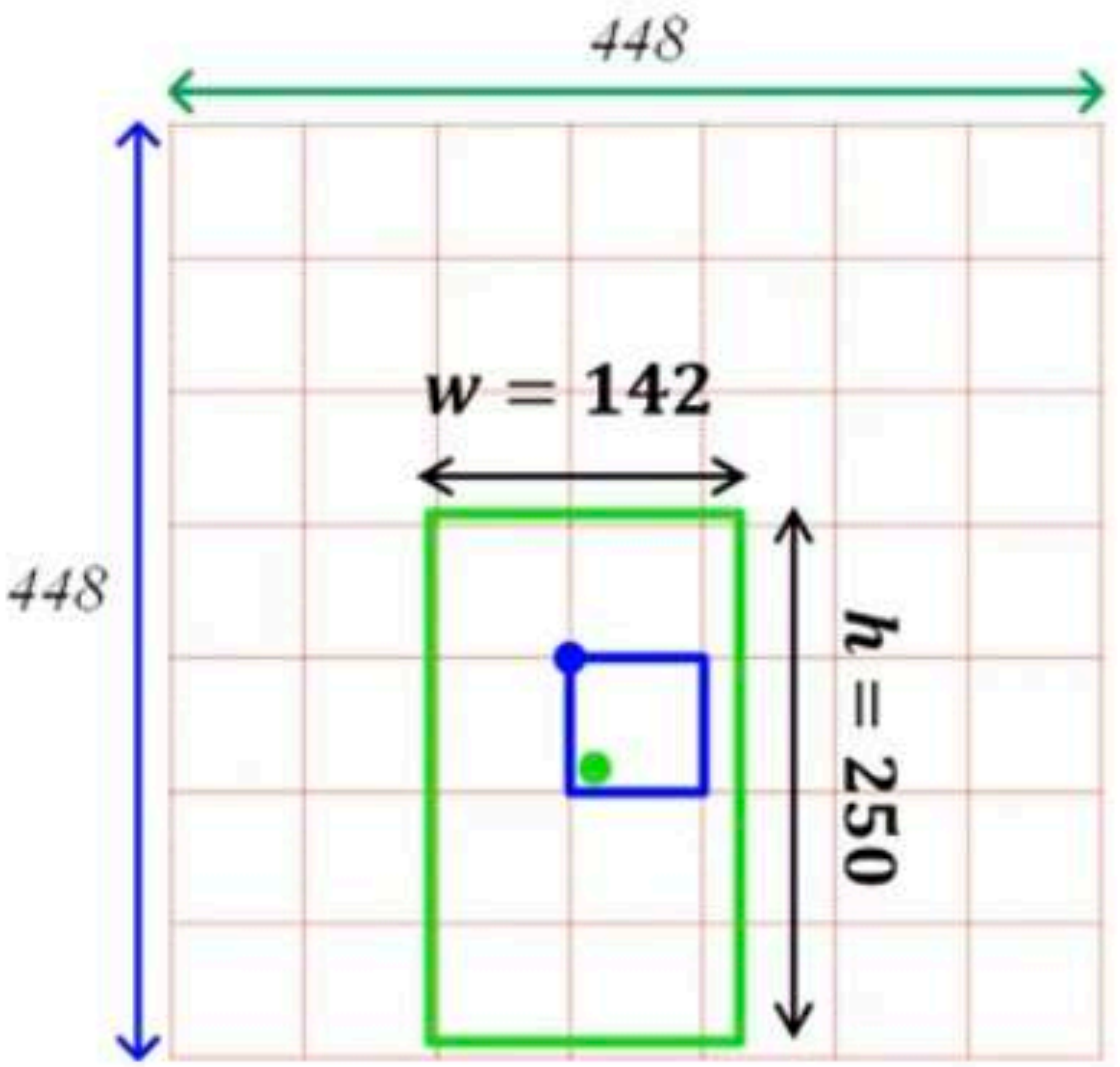
$$\Delta x = \frac{200 - 192}{64} \approx 0.13$$

$$\Delta y = \frac{311 - 256}{64} \approx 0.87$$

$$\Delta w = w/448$$

$$\Delta h = h/448$$

Example Calculation - Targets



$$\Delta x = \frac{200 - 192}{64} \approx 0.13$$

$$\Delta y = \frac{311 - 256}{64} \approx 0.87$$

$$\Delta w = \frac{142}{448} \approx 0.31$$

$$\Delta h = \frac{250}{448} \approx 0.56$$

$$(200, 311, 142, 250) \Rightarrow \underline{(0.13,\ 0.87,\ 0.31,\ 0.56)}$$

Label Encoding

- For every Grid cell (Anchor box), we need to create targets/labels
- No Object - All zeros
- Object - Relative values w.r.t grid
- Classes - one-hot encoding
- Ex: (x,y,w,h,c) = (0.9,0.7,0.1,0.1,1.0)
  - Classes = (1.0, 0,0,…0) - 20 values

	$(\Delta\hat{x},\ \Delta\hat{y},\ \Delta\hat{w},\ \Delta\hat{h},\ \hat{c})$	$(\hat{p}_1,\ \hat{p}_2,\ \dots,\ \hat{p}_{20})$
A_1	$(0\ \ 0\ \ 0\ \ 0\ \ 0)$	$(\ 0\ \ 0\ \ \dots\ \ 0\ )$
A_2	$(0\ \ 0\ \ 0\ \ 0\ \ 0)$	$(\ 0\ \ 0\ \ \dots\ \ 0\ )$
	…………… .	
A_{11}	$(0.9\ \ 0.7\ \ 0.1\ \ 0.1\ \ 1.0)$	$(\ 0\ \ \dots\ \ 1.0\ \ \dots\ )$
	…………… .	$\hat{p}_{14=\text{person}}$
A_{32}	$(0.1\ \ 0.8\ \ 0.3\ \ 0.5\ \ 1.0)$	$(\ 0\ \ \dots\ \ 1.0\ \ \dots\ )$
	…………… .	
A_{49}	$(0\ \ 0\ \ 0\ \ 0\ \ 0)$	$(\ 0\ \ 0\ \ \dots\ \ 0\ )$

Prediction Vector

- Each grid cell (we consider that as anchor here) predicts:
  - 2 Bounding boxes (B=2)
  - For each bounding box:

$$(\Delta x_i, \Delta y_i, \Delta w_i, \Delta h_i, c_i)_{i=1}^{B}$$

  - Conditional class probabilities (n=20)
    - (P1,P2,…,P20)

Prediction Vector

- Each grid cell (we consider that as anchor here) predicts:
  - 2 Bounding boxes (B=2)
  - For each bounding box:

Offsets relative to top-left corner of grid cell

$$(\boxed{\Delta x_i, \Delta y_i}, \Delta w_i, \Delta h_i, c_i)_{i=1}^{B}$$

  - Conditional class probabilities (n=20)
    - (P1,P2,…,P20)

Prediction Vector

- Each grid cell (we consider that as anchor here) predicts:
  - 2 Bounding boxes (B=2)
  - For each bounding box:

Offsets relative to width and height of image

$$(\Delta x_i, \Delta y_i, \boxed{\Delta w_i, \Delta h_i,} c_i)_{i=1}^{B}$$

  - Conditional class probabilities (n=20)
    - (P1,P2,…,P20)

Prediction Vector

- Each grid cell (we consider that as anchor here) predicts:
  - 2 Bounding boxes (B=2)
  - For each bounding box:

Probability that the box contains an Object

$$(\Delta x_i, \Delta y_i, \Delta w_i, \Delta h_i, \boxed{c_i})_{i=1}^{B}$$

  - Conditional class probabilities (n=20)
    - (P1,P2,…,P20)

Prediction Vector

- Each grid cell (we consider that as anchor here) predicts:
  - 2 Bounding boxes (B=2)
  - For each bounding box:

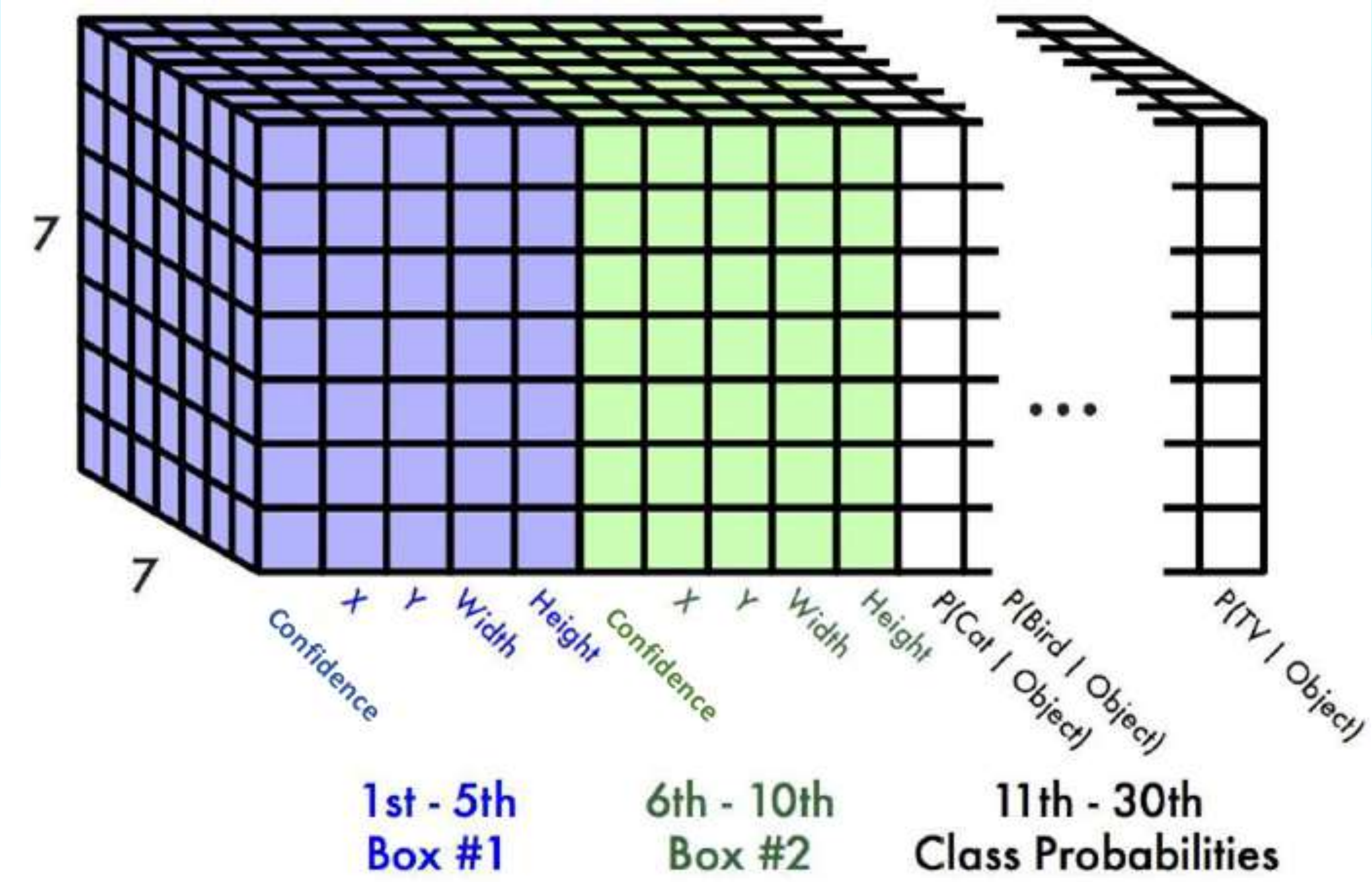
Probability they the box contains an Object

$$(\Delta x_i, \Delta y_i, \Delta w_i, \Delta h_i, \boxed{c_i})_{i=1}^{B}$$

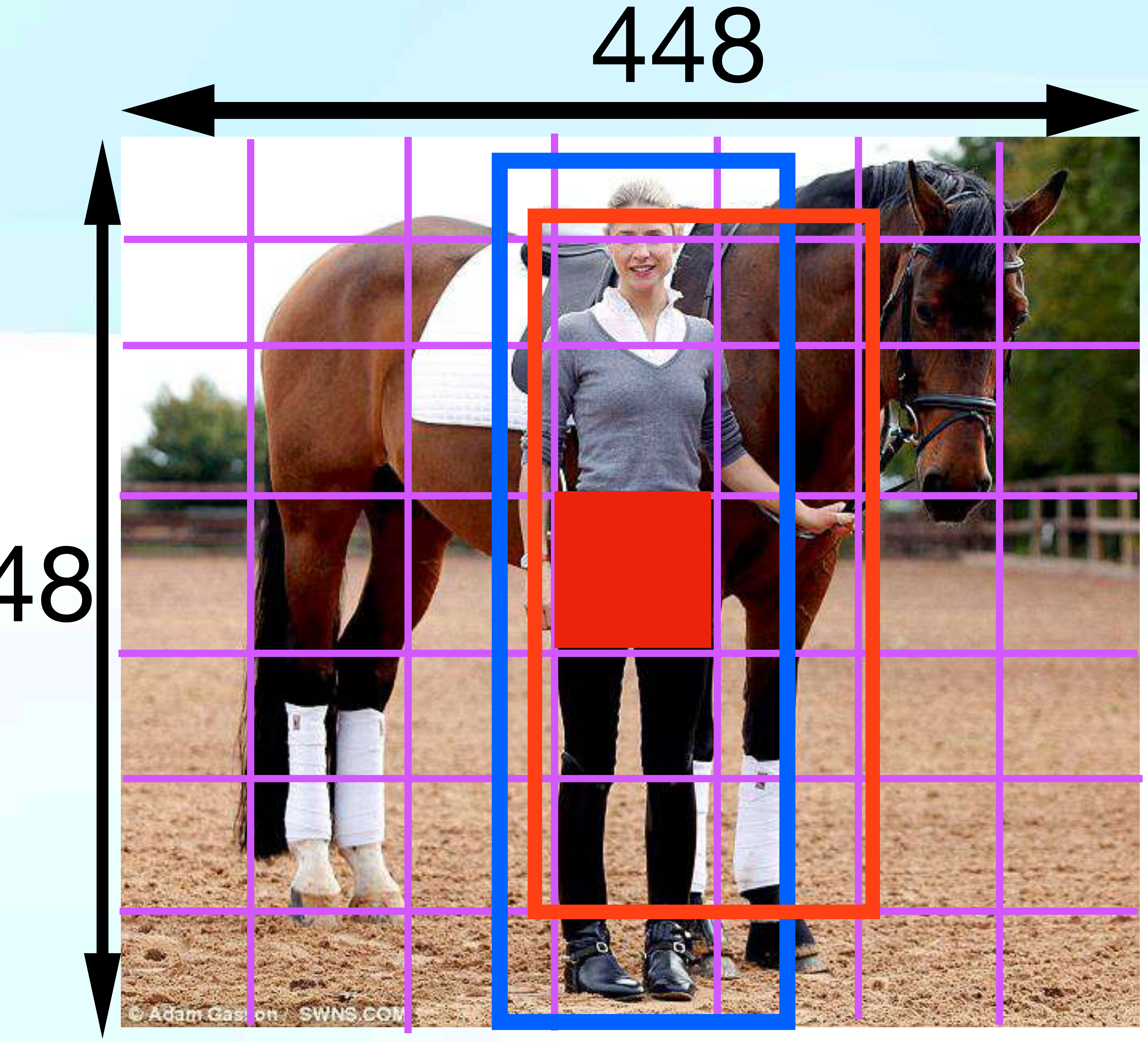
  - Conditional class probabilities (n=20)
    - (P1,P2,…,P20)
  - Number of parameters per grid cell: ((2x5) + 20 = 30)

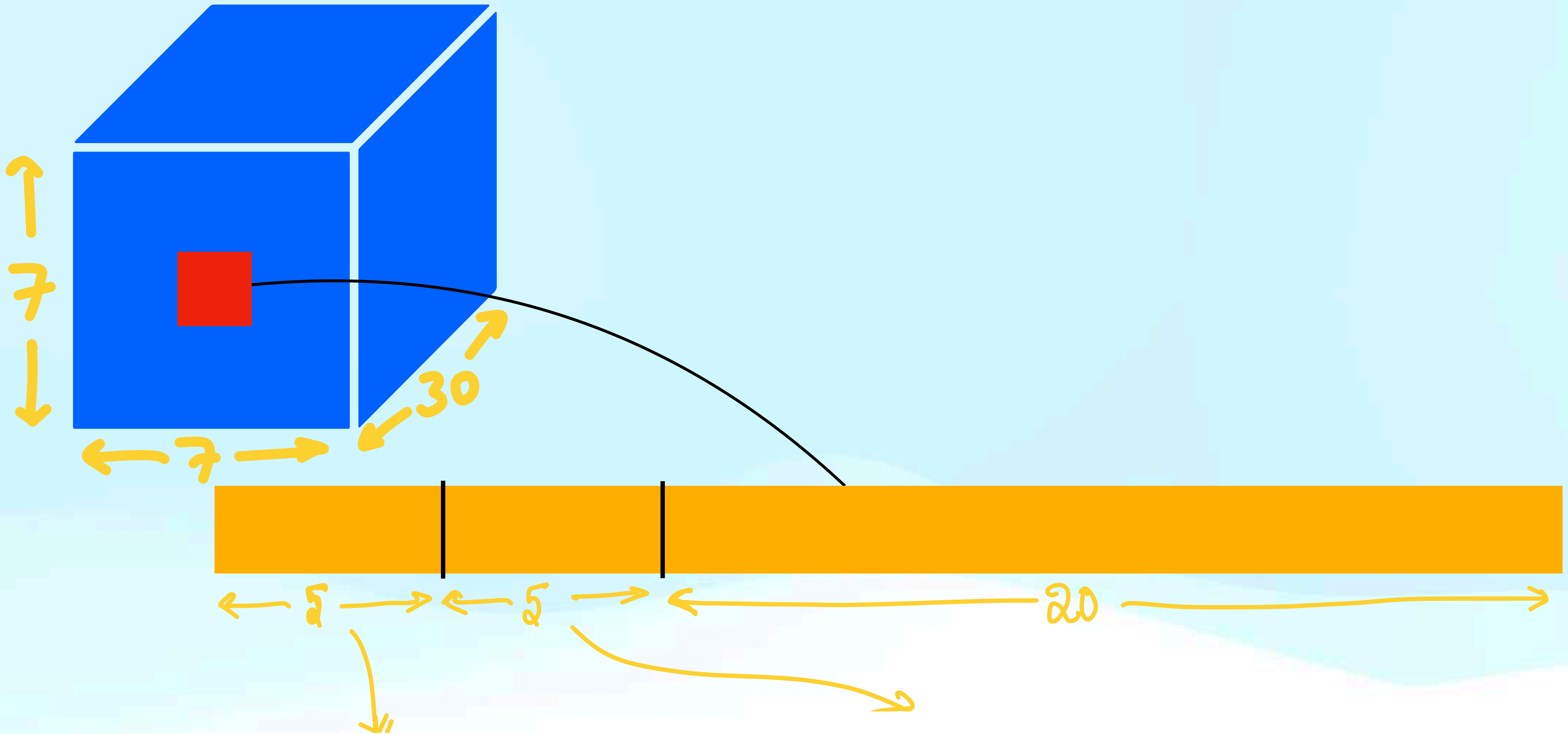
Prediction Vector

- Output layer - 7x7x30



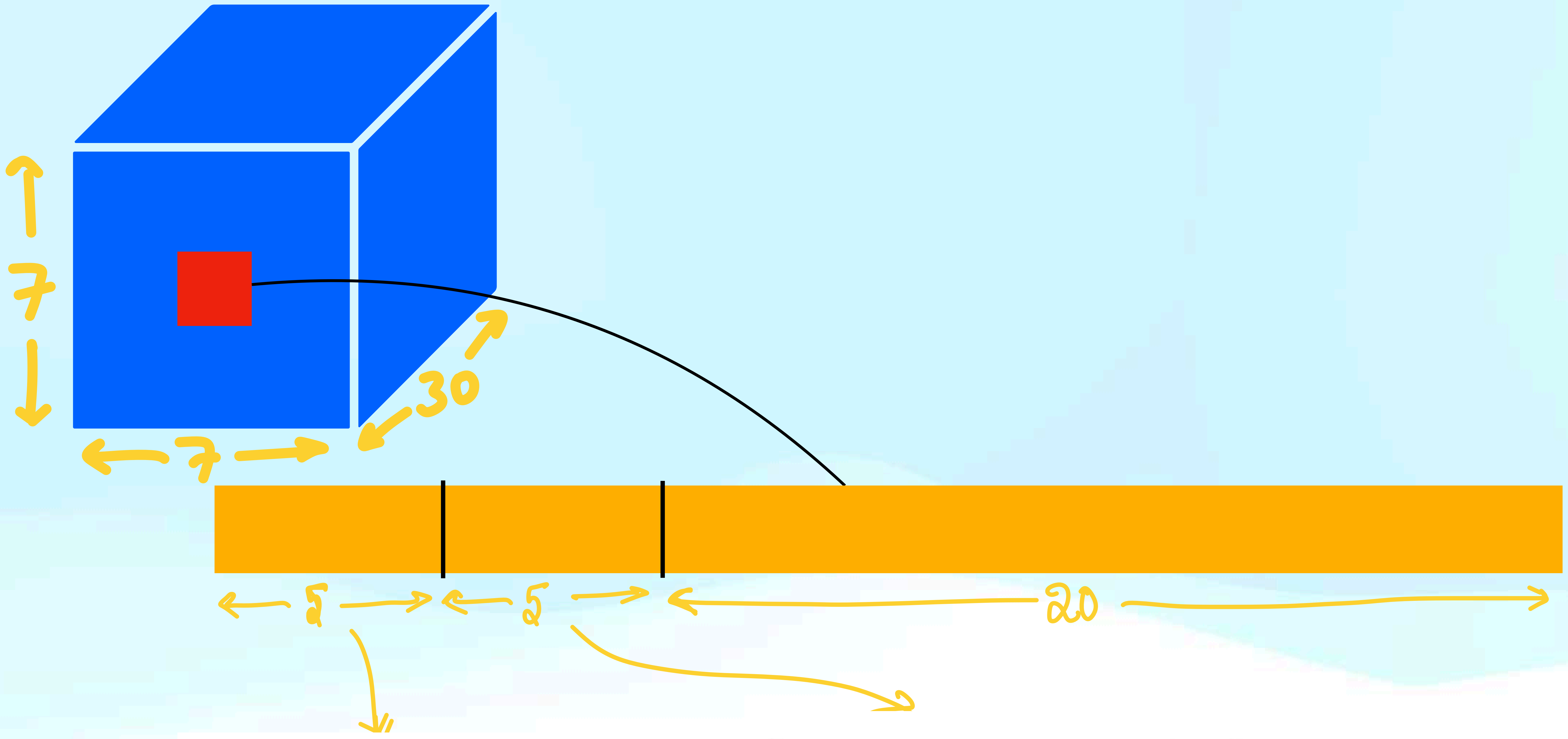
Output Parsing



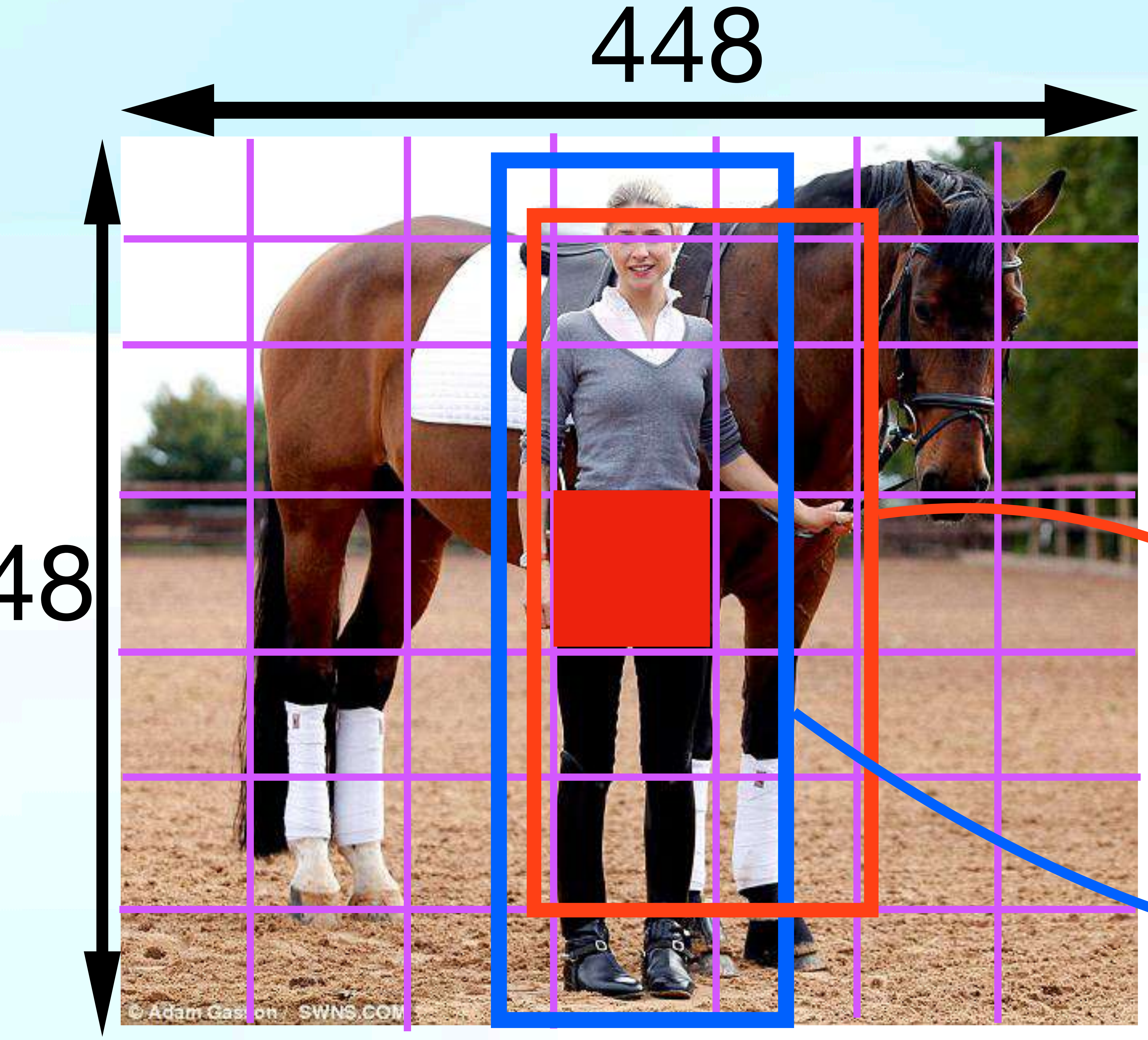


$(\Delta x_1, \Delta y_1, \Delta w_1, \Delta h_1)$,C1 $(\Delta x_2, \Delta y_2, \Delta w_2, \Delta h_2)$,C2

Output Parsing



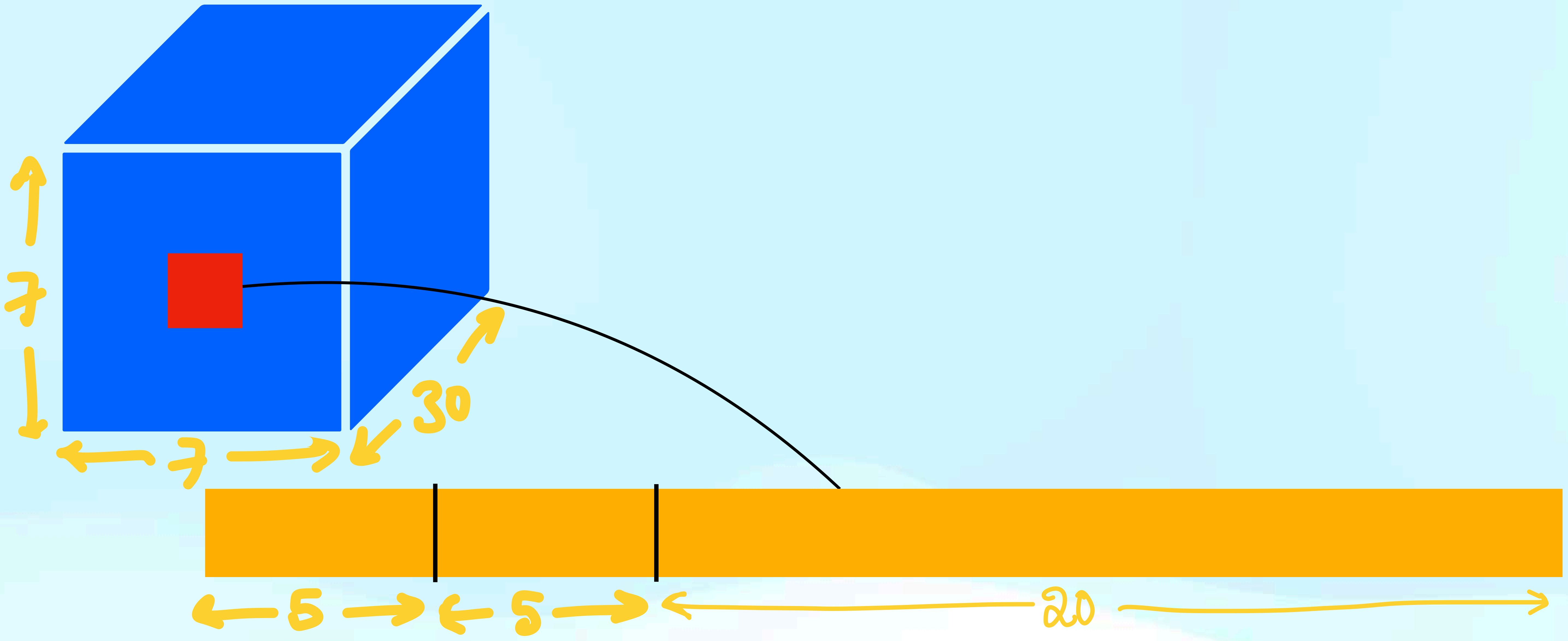
$(\Delta x_1, \Delta y_1, \Delta w_1, \Delta h_1)$,C1 $(\Delta x_2, \Delta y_2, \Delta w_2, \Delta h_2)$,C2

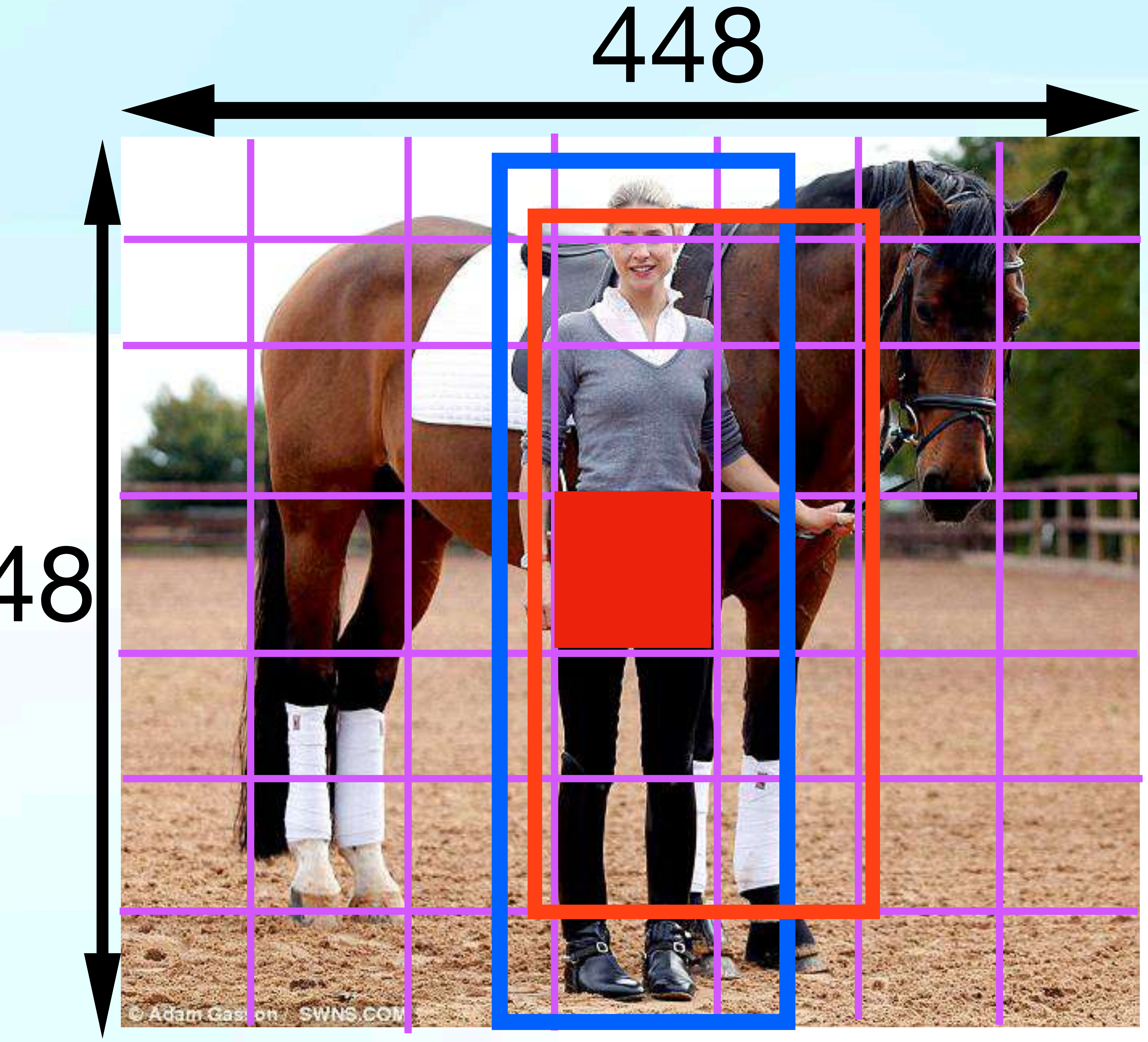


$$\begin{cases} x_1 = \Delta x_1 \times 64 + x_a \\ y_1 = \Delta y_1 \times 64 + y_a \\ w_1 = \Delta w_1 \times 448 \\ h_1 = \Delta h_1 \times 448 \end{cases}$$

$$\begin{cases} x_2 = \Delta x_2 \times 64 + x_a \\ y_2 = \Delta y_2 \times 64 + y_a \\ w_2 = \Delta w_2 \times 448 \\ h_2 = \Delta h_2 \times 448 \end{cases}$$

Output Parsing



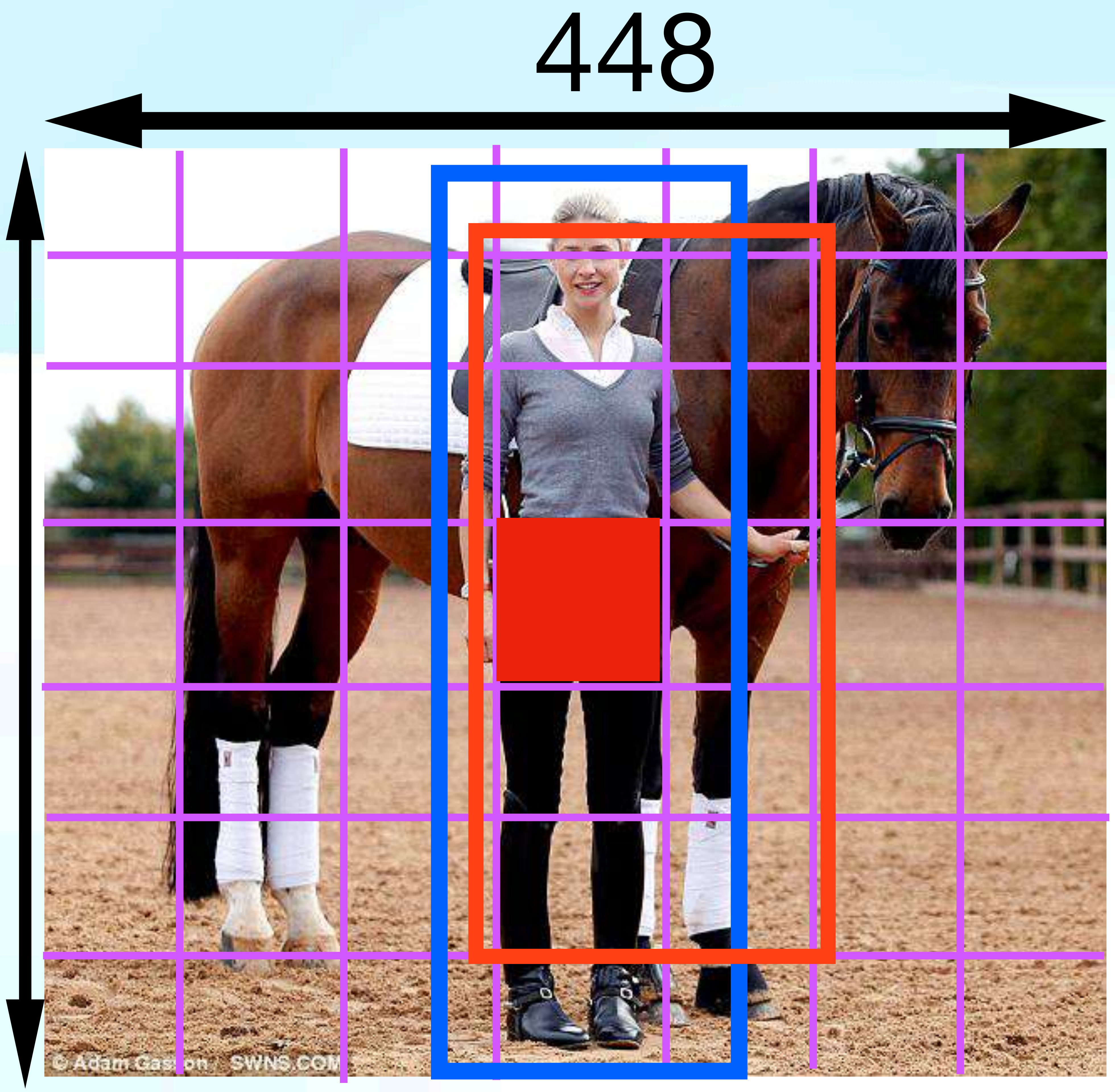


object class $l = \operatorname{argmax}(p_1, p_2, \ldots, p_{20})$

class confidence $\hat{c}_1 = c_1 \times p$ $\hat{c}_2 = c_2 \times p$

$$p = \max(p_1, p_2, \ldots, p_{20})$$

Post processing



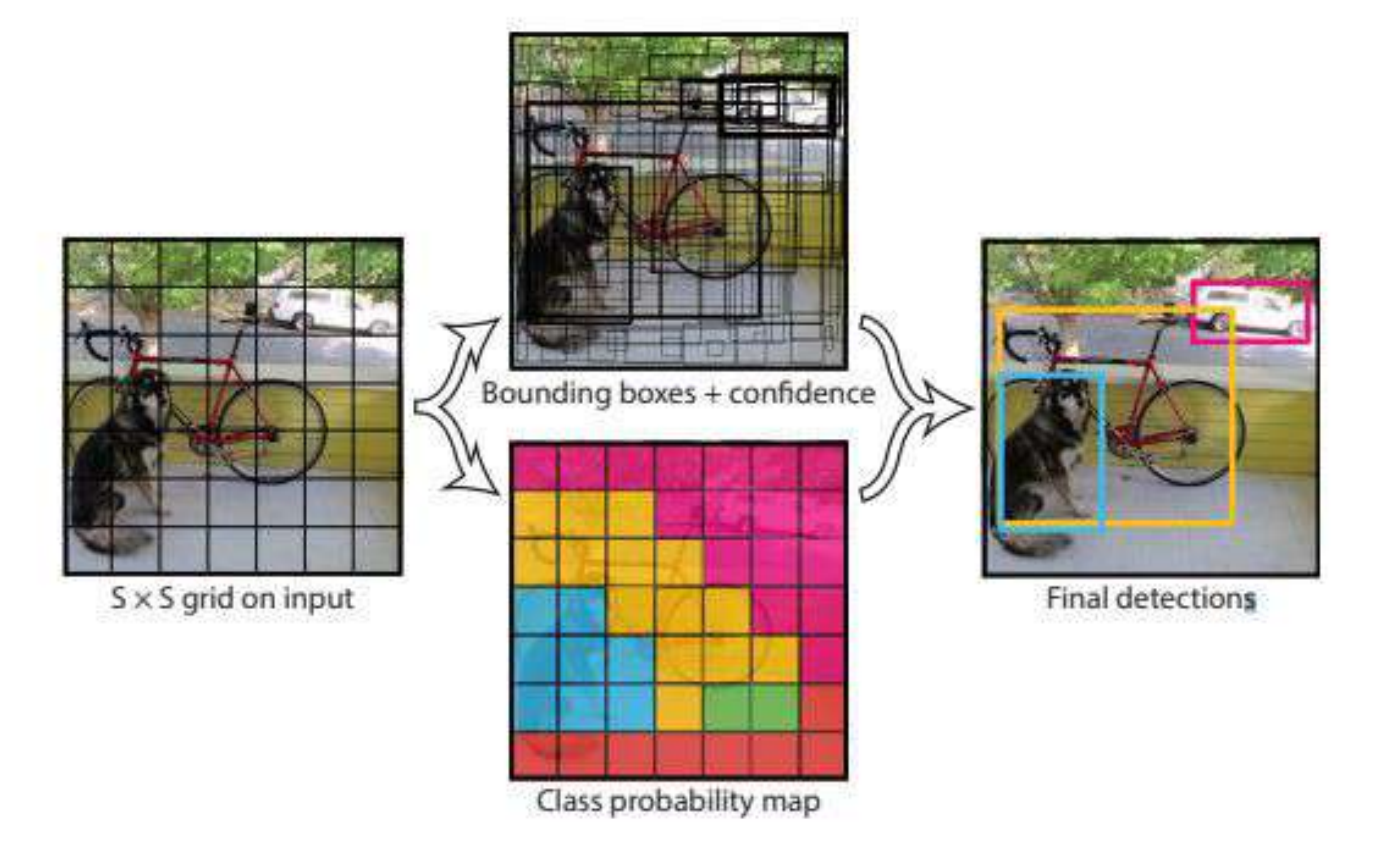
object class $l = \text{argmax}(p_1, p_2, \ldots, p_{20})$

class confidence $\hat{c}_1 = c_1 \times p$ $\hat{c}_2 = c_2 \times p$

$p = \max(p_1, p_2, \ldots, p_{20})$

* Consider the box with highest confidence score per each grid

Overview - so far



Architecture

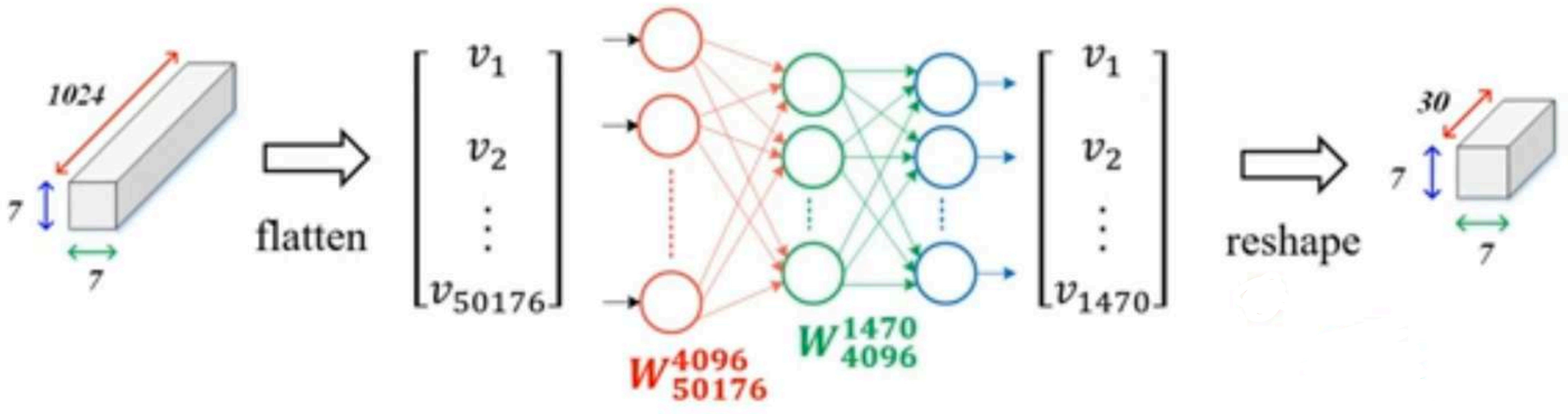
- Inspired by GoogleNet model
- Network:
    - 24 convolution all layers
    - 2 fully connected layers

	Type	Size	Filters	Stride	Output	
	Conv.	7 x 7 x 3	64	2	224 x 224 x 64	
	max pool	2 x 2			112 x 112 x 64	
	Conv.	3 x 3 x 64	192	1	112 x 112 x 192	
	max pool	2 x 2			56 x 56 x 192	
	Conv.	1 x 1 x 192	128	1	56 x 56 x 128	6
	Conv.	3 x 3 x 128	256	1	56 x 56 x 256	
	Conv.	1 x 1 x 256	256	1	56 x 56 x 256	
	Conv.	3 x 3 x 256	512	1	56 x 56 x 512	
	max pool	2 x 2			28 x 28 x 512	
4 x	Conv.	1 x 1 x 512	256	1	28 x 28 x 256	$4 \times 2 = 8$
	Conv.	3 x 3 x 256	512	1	28 x 28 x 512	
	Conv.	1 x 1 x 512	512	1	28 x 28 x 512	
	Conv.	3 x 3 x 512	1024	1	28 x 28 x1024	2
	max pool	2 x 2			14 x 14 x1024	
2 x	Conv.	1 x 1 x 1024	512	1	14 x 14 x 512	$2 \times 2 = 4$
	Conv.	3 x 3 x 512	1024	1	14 x 14 x1024	
	Conv.	3 x 3 x 1024	1024	1	14 x 14 x1024	
	Conv.	3 x 3 x 1024	1024	2	7 x 7 x1024	4
	Conv.	3 x 3 x 1024	1024	1	7 x 7 x1024	
	Conv.	3 x 3 x 1024	1024	1	7 x 7 x1024	

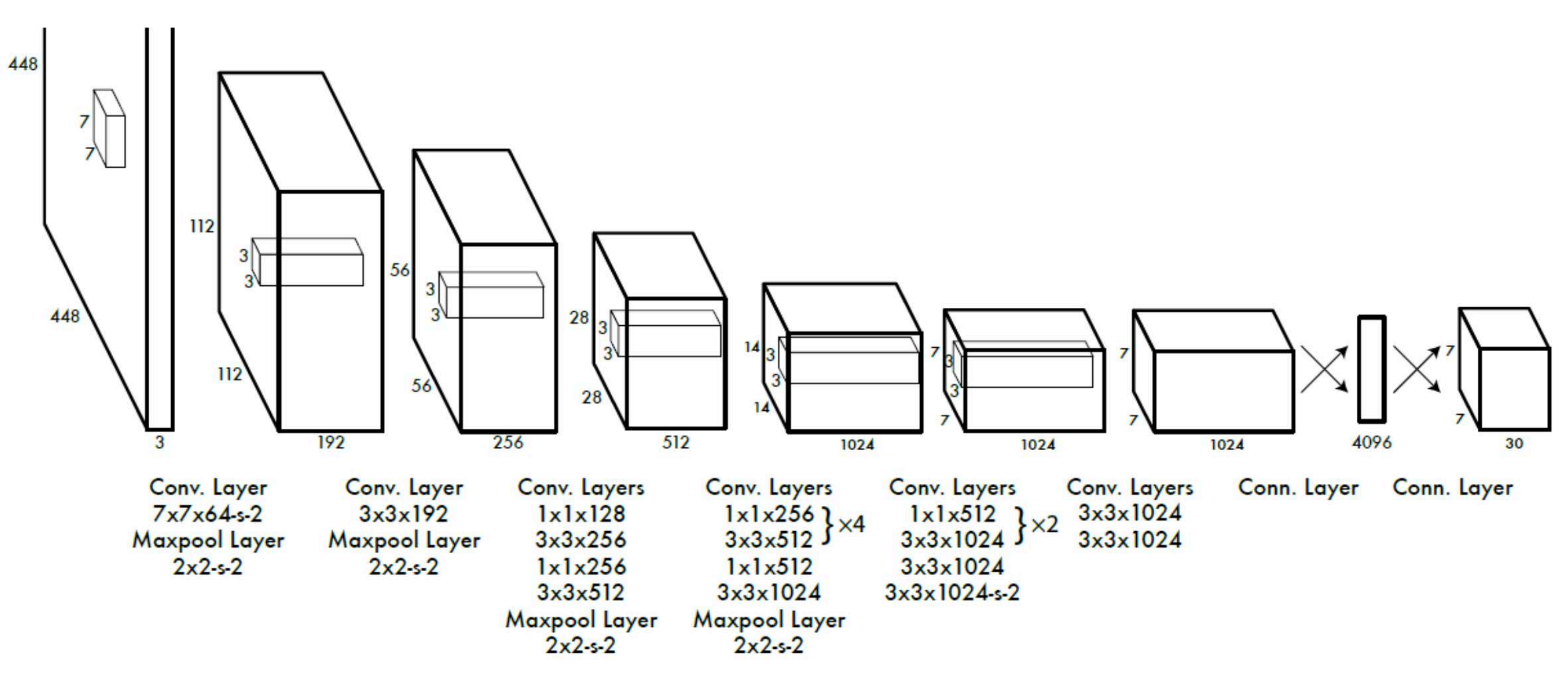
$$6 + 8 + 2 + 4 + 4 = 24$$

Architecture

- Flatten the last conv map 7x7x1024 to 50176 feature vector
- Pass through 2 fully connected layers
- Output - 1470 feature vector
- Reshape 1470 vector to 7x7x30 feature map



YOLO Architecture

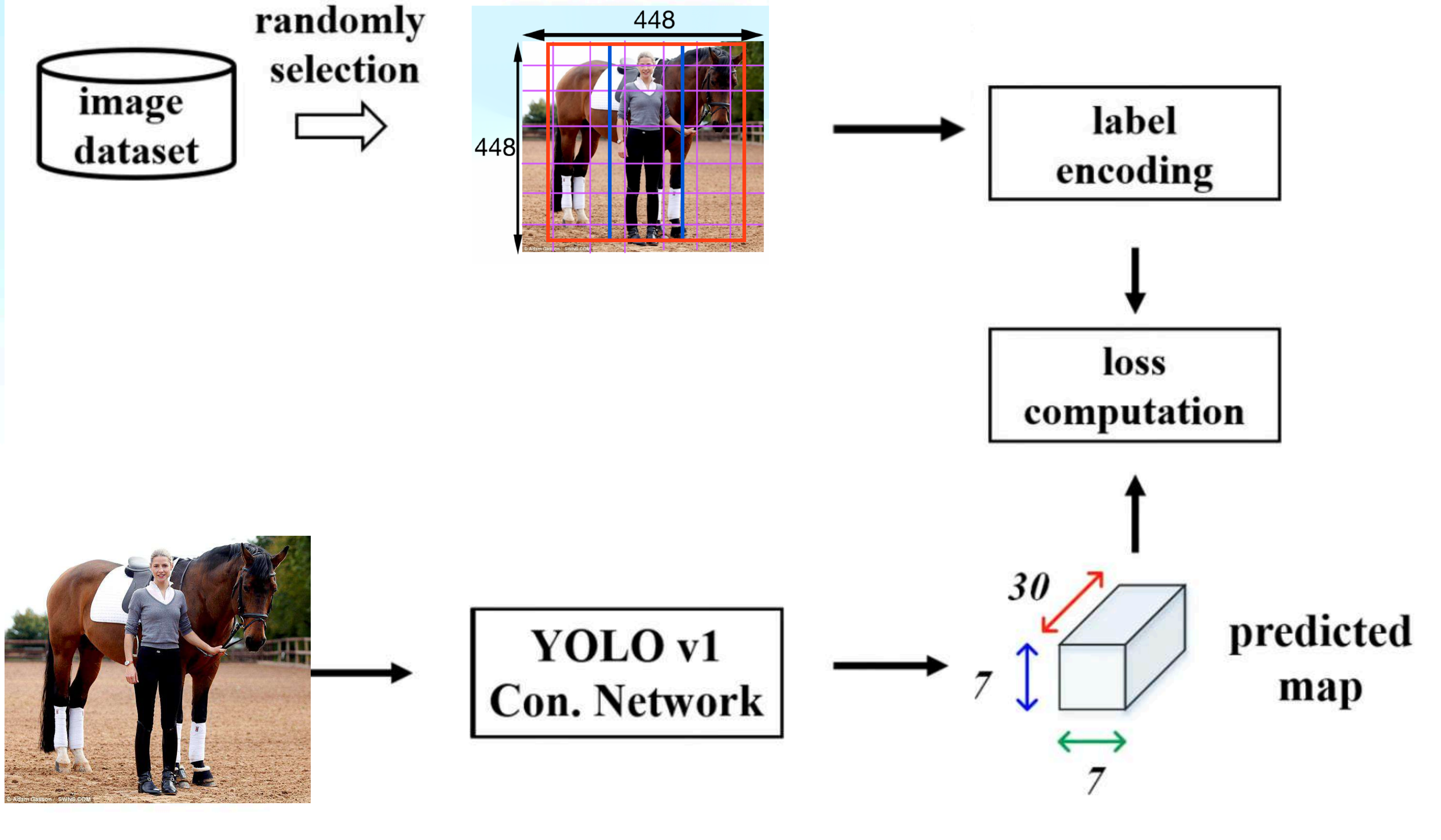


Training Process

- Dataset: Pascal VOC - 20 classes
- Network pretrained on Imagenet at 224x224
- Actual training on 448x448 on VOC dataset

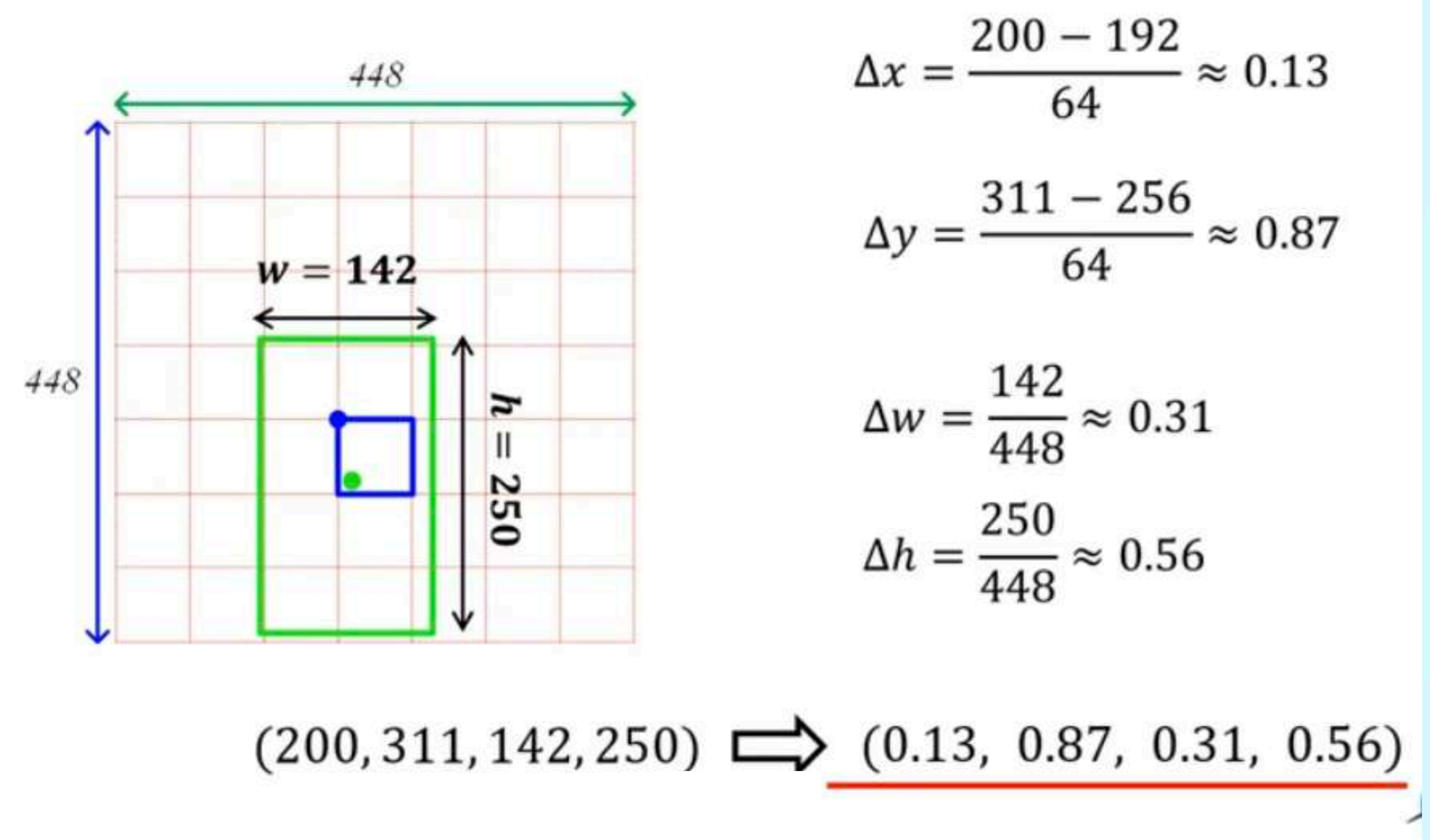
Training Process

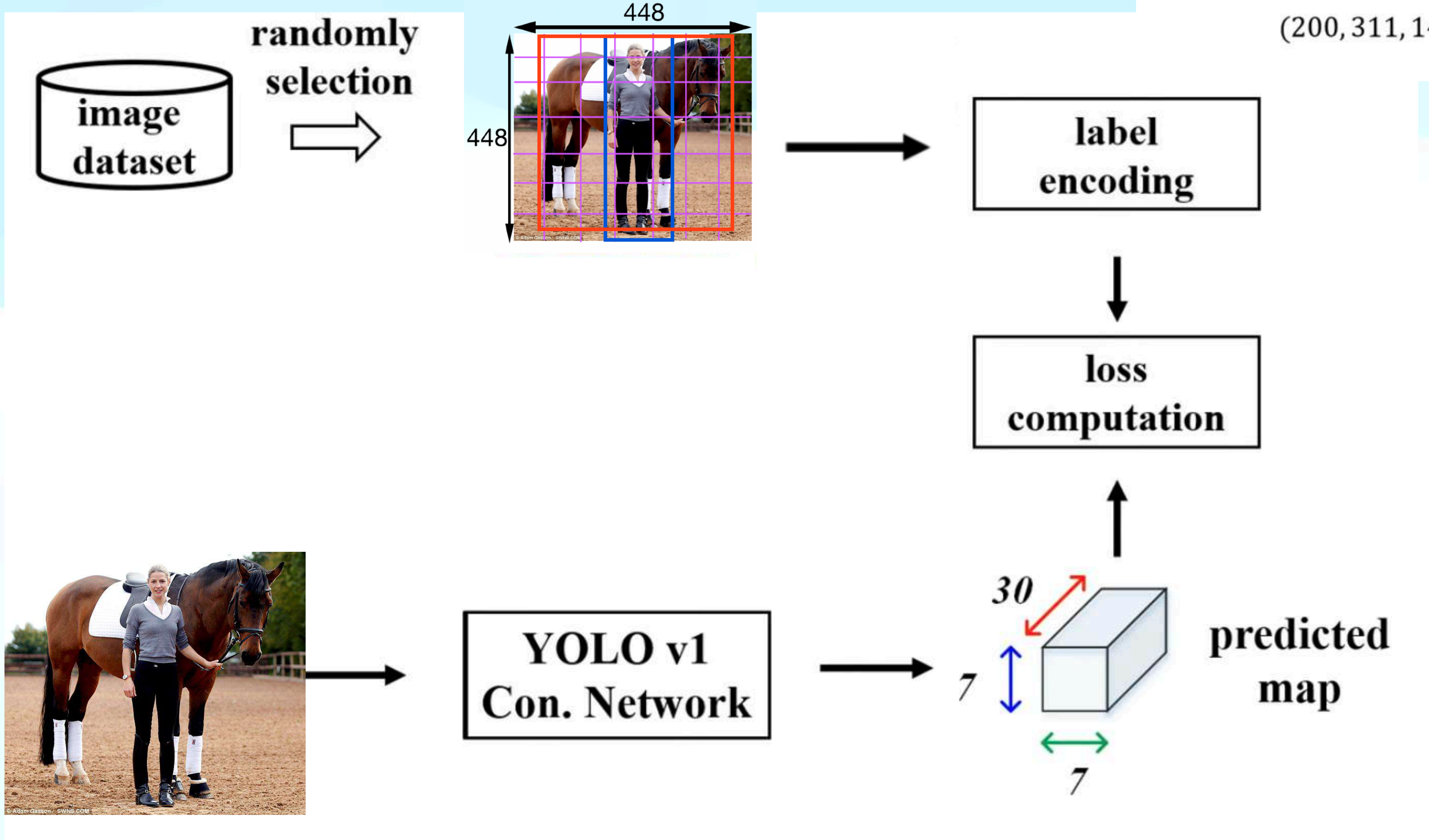
- Dataset: Pascal VOC - 20 classes



Training Process

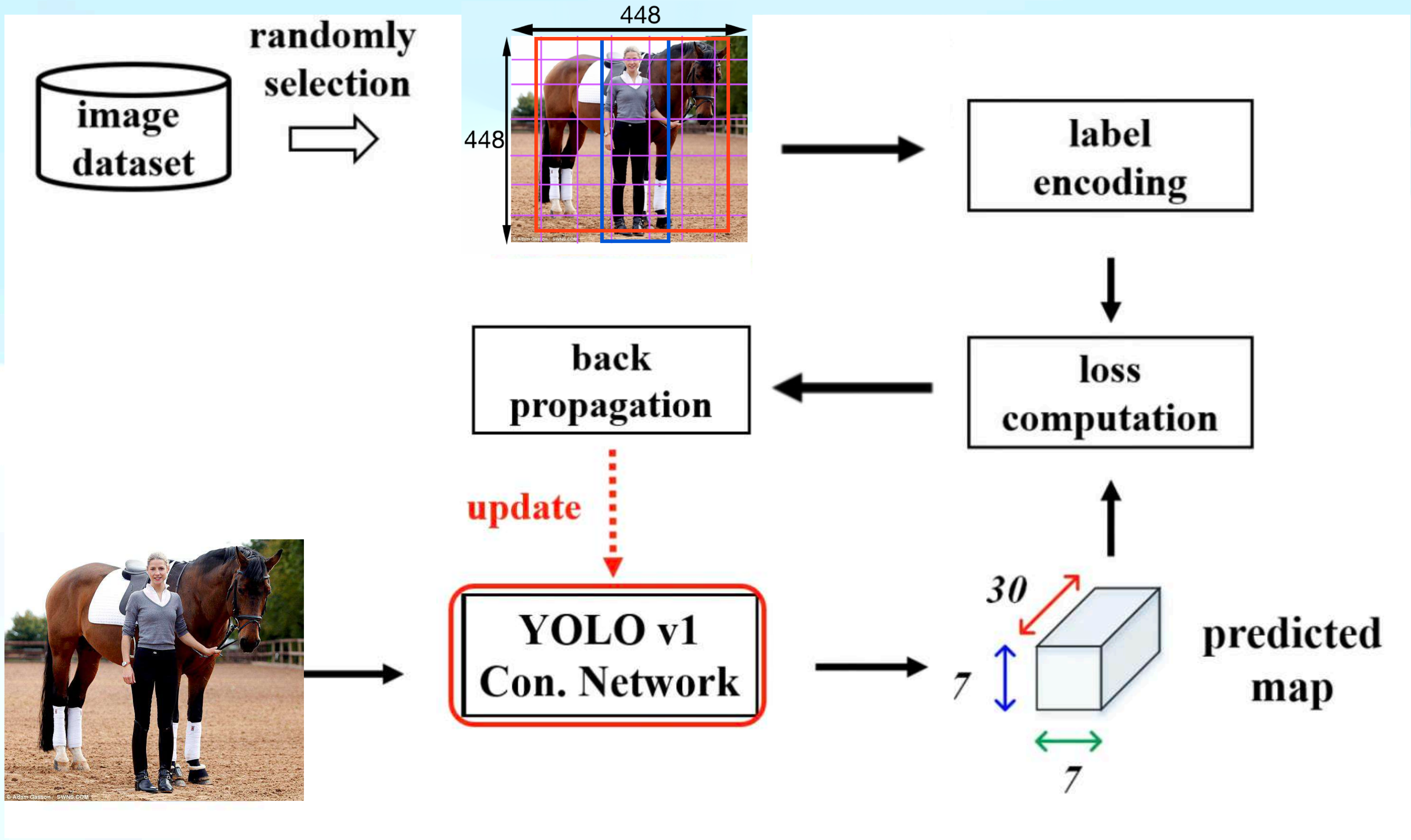
- Dataset: Pascal VOC - 20 classes





Training Process

- Dataset: Pascal VOC - 20 classes



Loss Function

- Loss L is the sum of losses over all grid cells SxS.

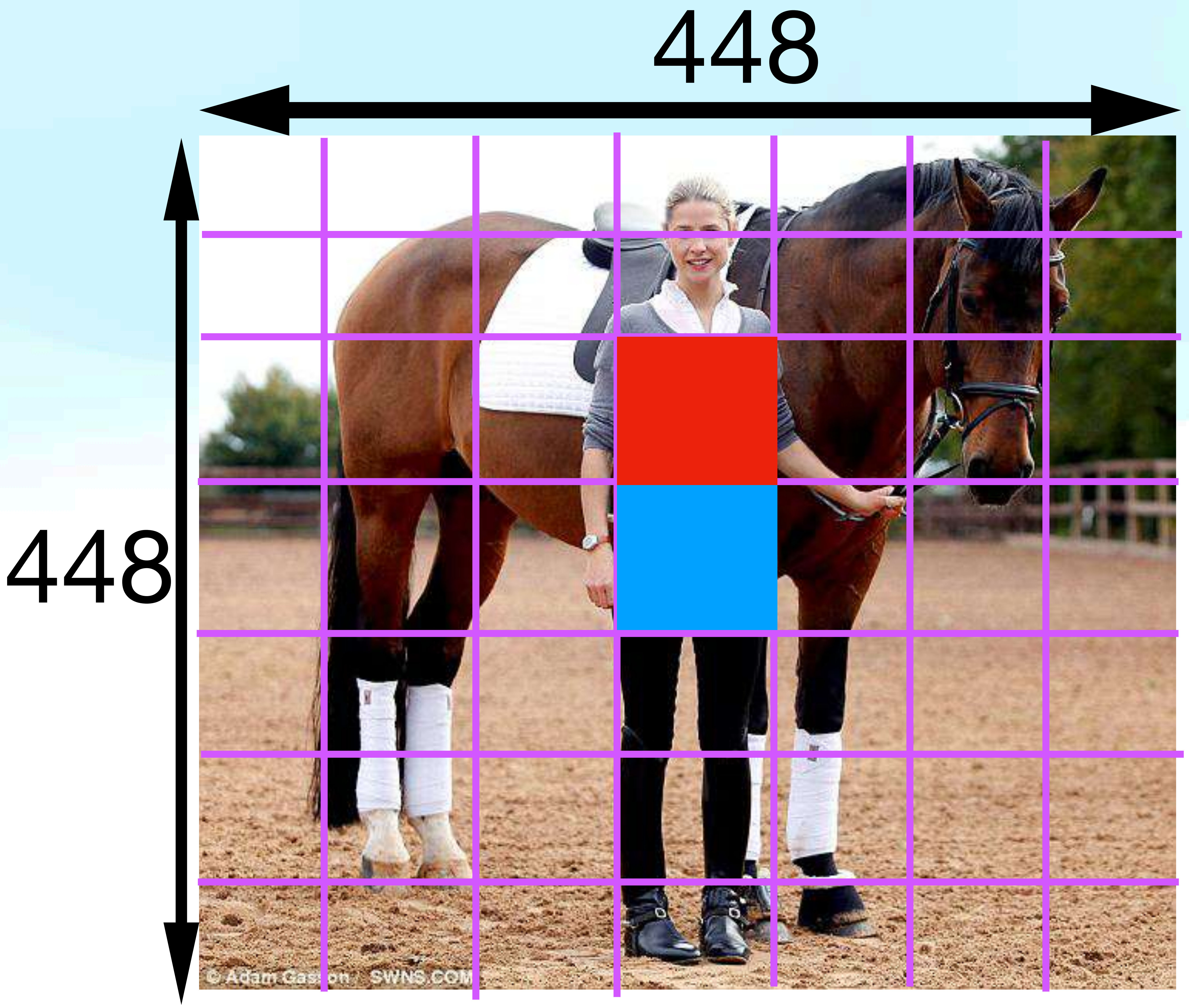
$$L = \sum_{i=1}^{S^2} \boxed{L_i}$$

Loss for i-th grid cell

Loss Function

- Loss L is the sum of losses over all grid cells SxS.
- Put more importance on grid cells that contain objects
- Decrease the importance of grid cells having no objects
- Ex: 2 object cells, 47 no-object cells

$$\boldsymbol{L} = \sum_{i=1}^{S^2} L_i$$



Loss Function

- Loss L is the sum of losses over all grid cells SxS.
- Put more importance on grid cells that contain objects
- Decrease the importance of grid cells having no objects

$$\boldsymbol{L} = \sum_{i=1}^{S^2} 1_i^{obj} \times L_{i,obj} + \underbrace{\boxed{\lambda_{no\_obj}}}_{= 0.5} \sum_{i=1}^{S^2} 1_i^{no\_obj} \times L_{i,no\_obj}$$

Loss Function

- Loss L is the sum of losses over all grid cells SxS.
- Put more importance on grid cells that contain objects
- Decrease the importance of grid cells having no objects

$$\boldsymbol{L} = \sum_{i=1}^{S^2} \boxed{1_i^{obj}} \times L_{i,obj} + \lambda_{no\_obj} \sum_{i=1}^{S^2} \boxed{1_i^{no\_obj}} \times L_{i,no\_obj}$$

$1_i^{obj} = 1$ if i^{th} grid is **object anchor**

$1_i^{no\_obj} = 1$ if i^{th} grid is **no-object anchor**

Loss for object cells

- Loss = Objectness loss + classification loss + Box Regression loss
- Put more weightage on box parameters

$$\boldsymbol{L_{i,obj}} = L_{i,obj}^{box} + L_{i,obj}^{conf} + L_{i,obj}^{cls}$$

Loss for object cells

- Loss = Objectness loss + classification loss + Box Regression loss
- Put more weightage on box parameters

$$\boldsymbol{L_{i,obj}} = \underbrace{\boxed{\lambda_{coord}}}_{=5} \times L_{i,obj}^{box} + L_{i,obj}^{conf} + L_{i,obj}^{cls}$$

Bounding box loss

- Sum of squared errors on predicted box parameters and ground truth labels

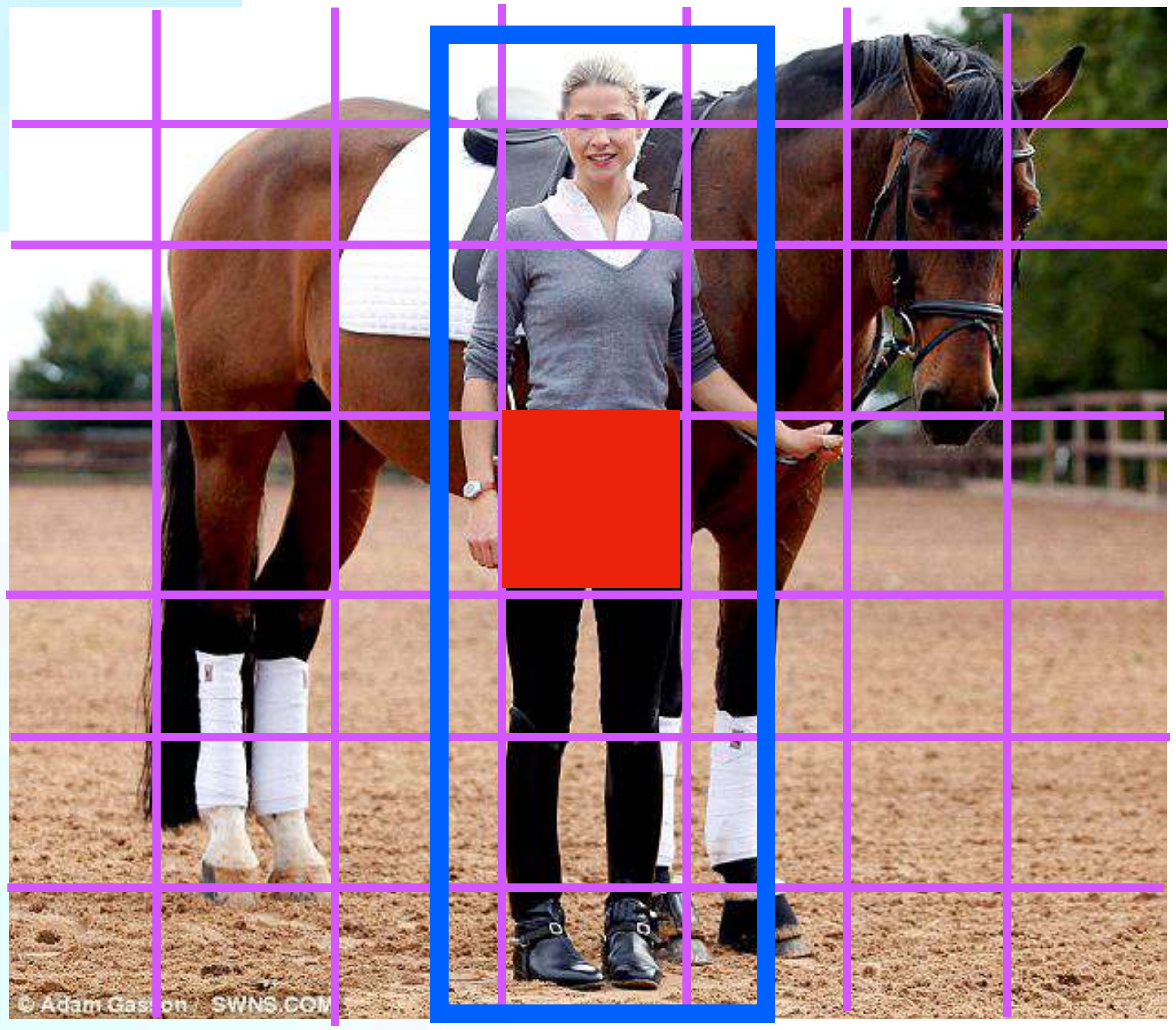
$$L_{i,obj}^{box} = \left(\Delta x_i^* - \Delta\hat{x}_i\right)^2 + \left(\Delta y_i^* - \Delta\hat{y}_i\right)^2$$
$$+ \left(\sqrt{\Delta w_i^*} - \sqrt{\Delta\widehat{w}_i}\right)^2 + \left(\sqrt{\Delta h_i^*} - \sqrt{\Delta h_i}\right)^2$$

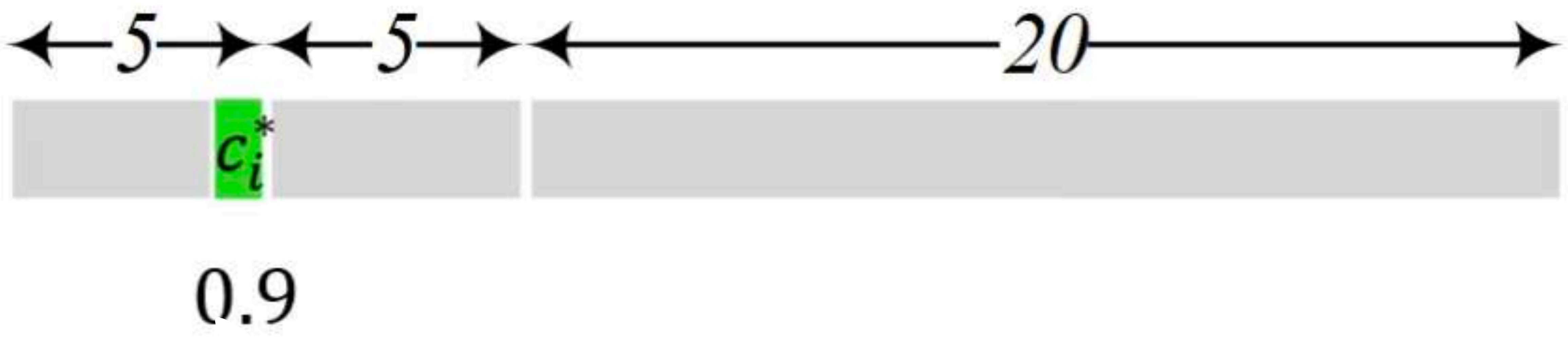
- $(\Delta\hat{x}_i, \Delta\hat{y}_i, \Delta\widehat{w}_i, \hat{h}_i)$: ground-truth box
- $(\Delta x_i^*, \Delta y_i^*, \Delta w_i^*, \Delta h_i^*)$: **responsible** predicted box that has the largest IoU with ground-truth box

Objectness Confidence Loss

- Squared error between the predicted confidence and encoded label confidence

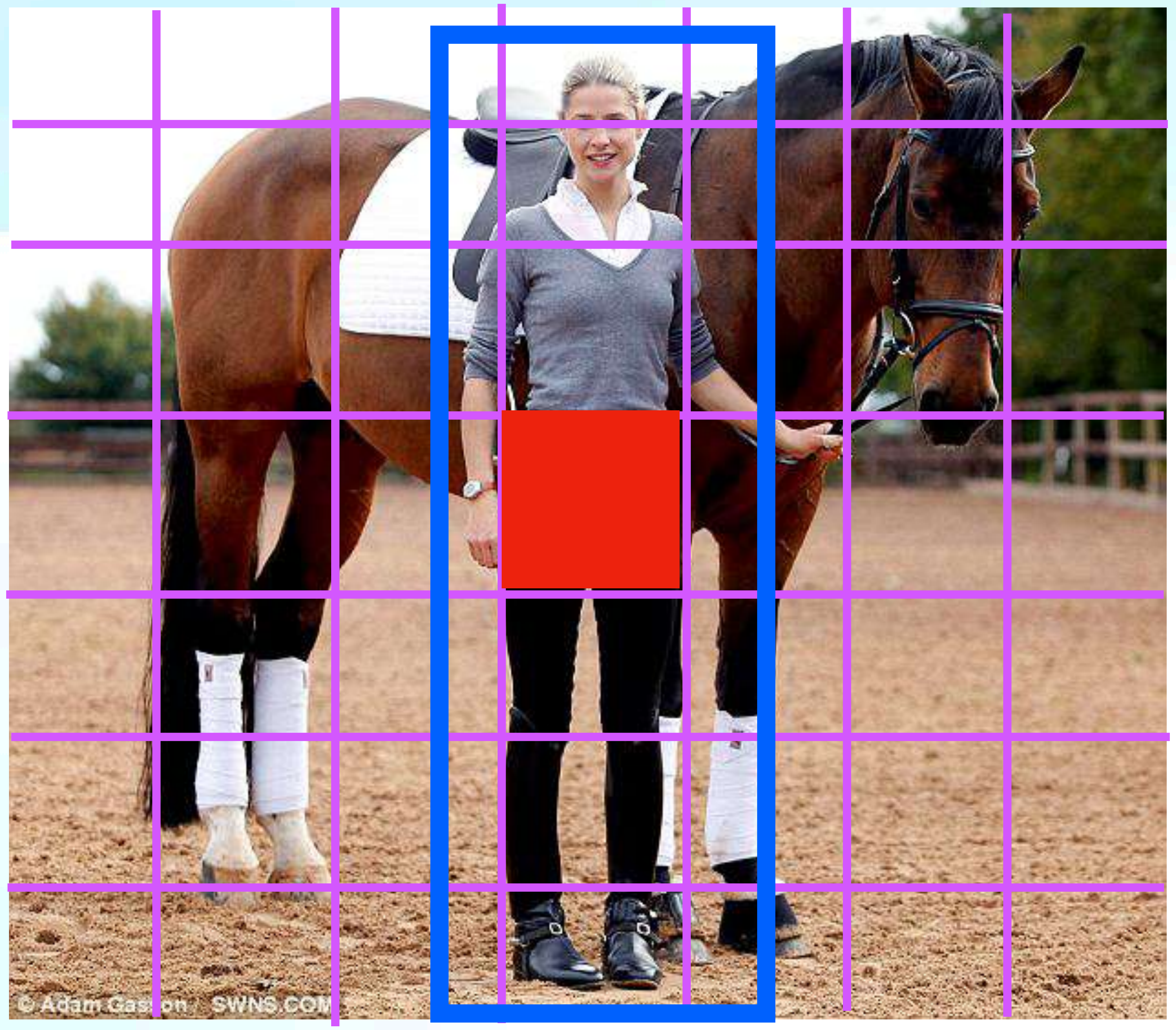
$$L_{i,obj}^{conf} = (c_i^* - \hat{c}_i)^2 = (0.9 - 1.0)^2$$





Classification Loss

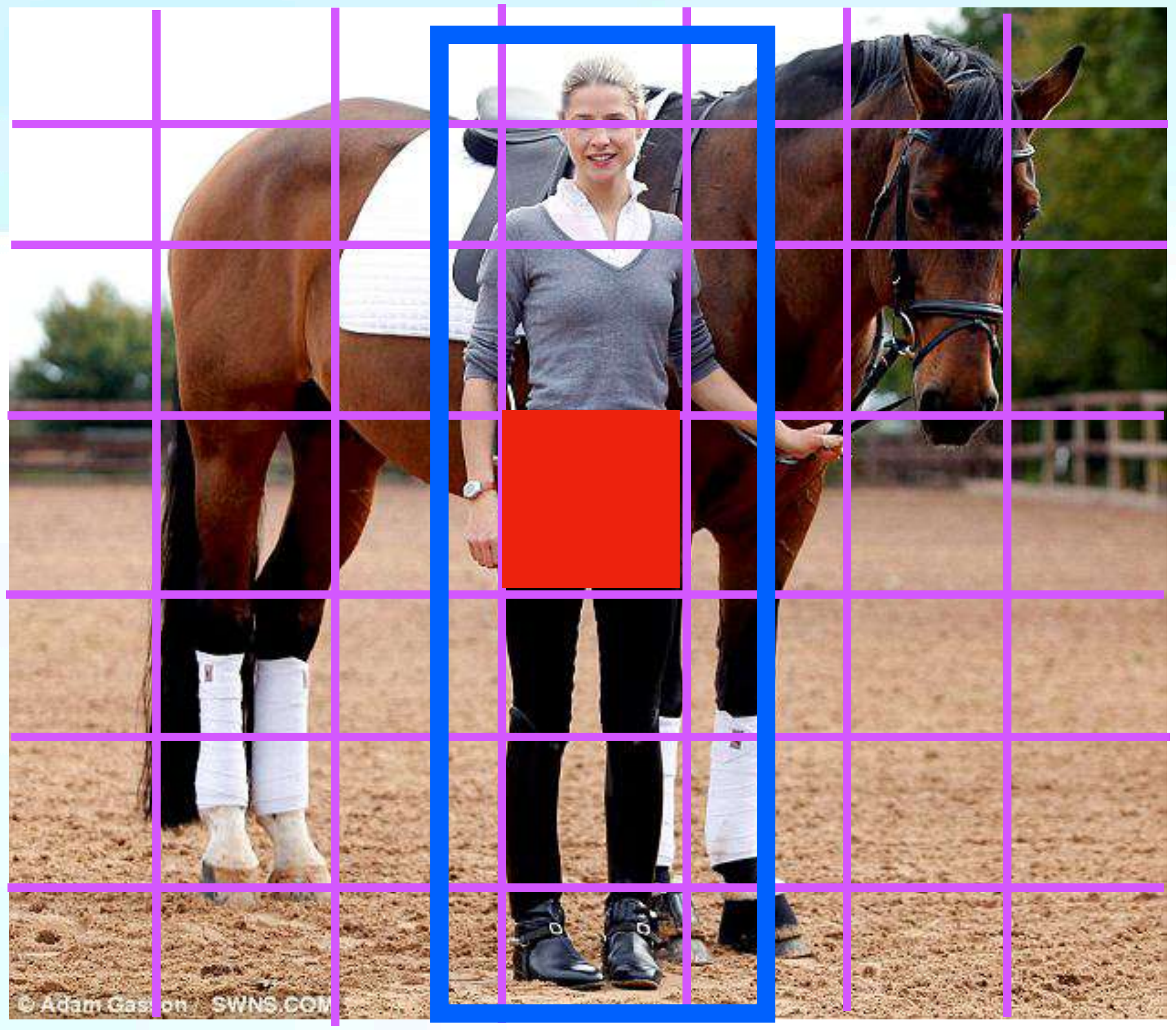
- Sum of squared errors over all class probabilities



$$L_{i,obj}^{cls} = \sum_{c=1}^{20} \left(p_{i,c} - \hat{p}_{i,c}\right)^2$$

Classification Loss

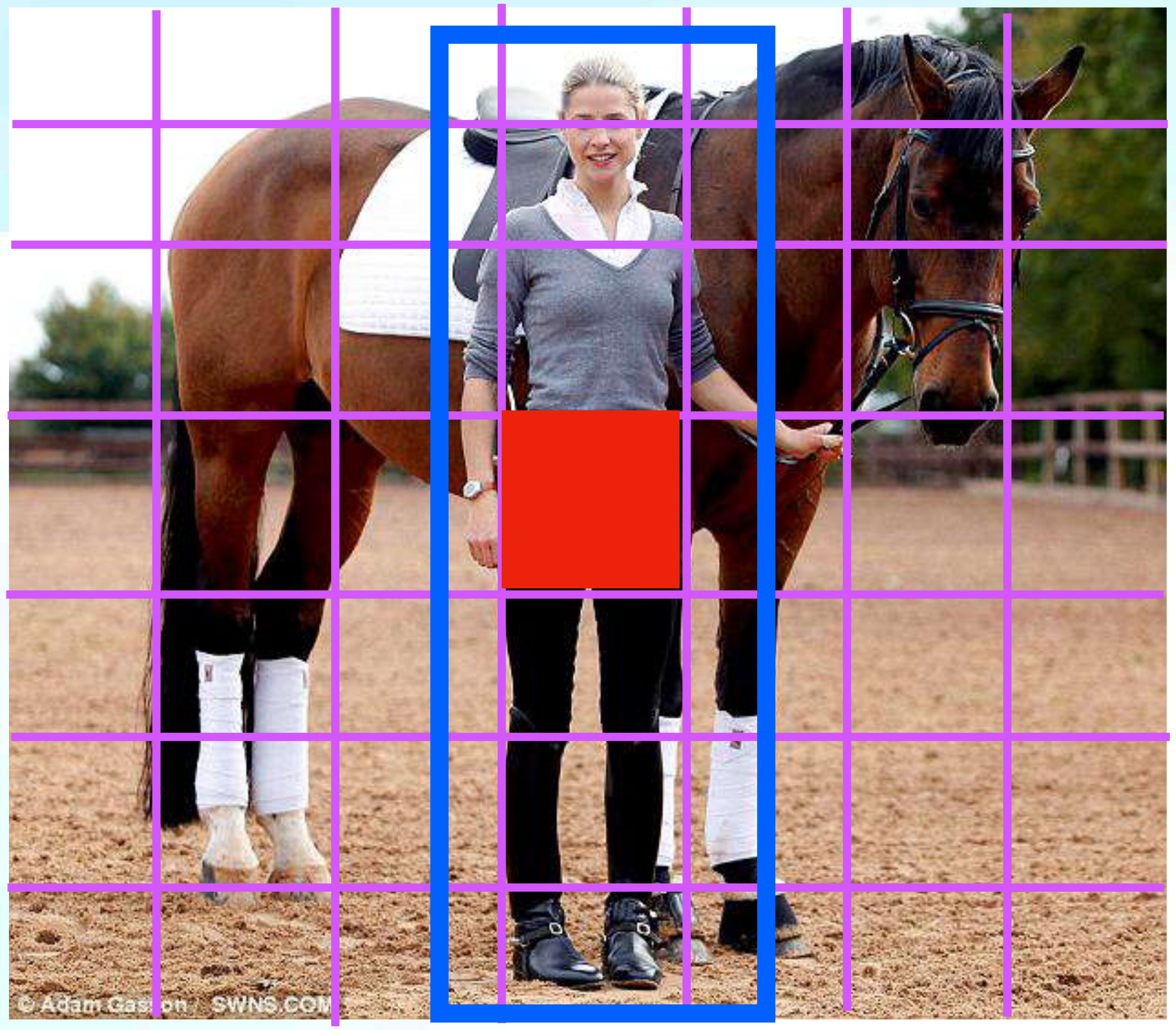
- Sum of squared errors over all class probabilities



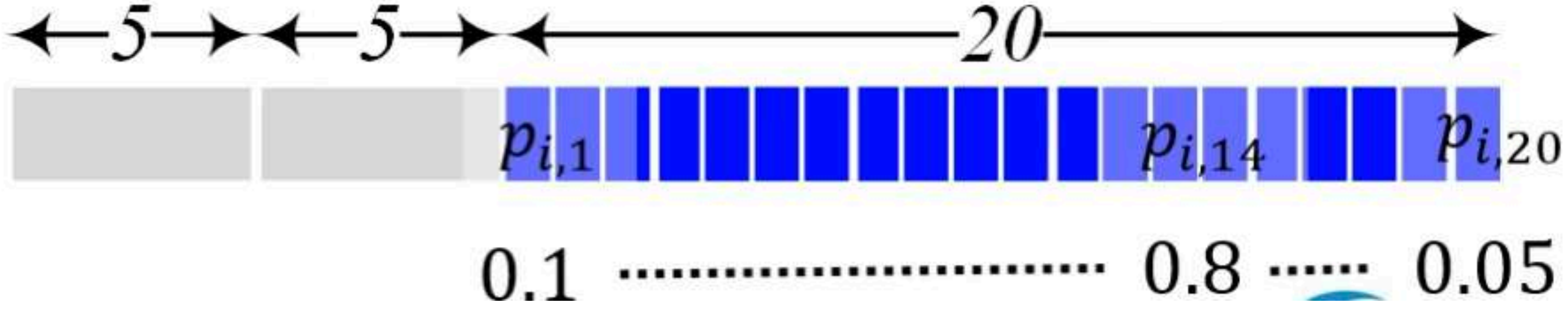
$$L^{cls}_{i,obj} = (p_{i,1} - \hat{p}_{i,1})^2 + \cdots + (p_{i,14} - \hat{p}_{i,14})^2 + \cdots + (p_{i,20} - \hat{p}_{i,20})^2$$

Classification Loss

- Sum of squared errors over all class probabilities



$$L^{cls}_{i,obj} = (0.1-0.0)^2 + \cdots + (0.8 - \overset{\hat{p}_{i,14}}{1.0})^2 + \cdots + (0.05-0.0)^2$$

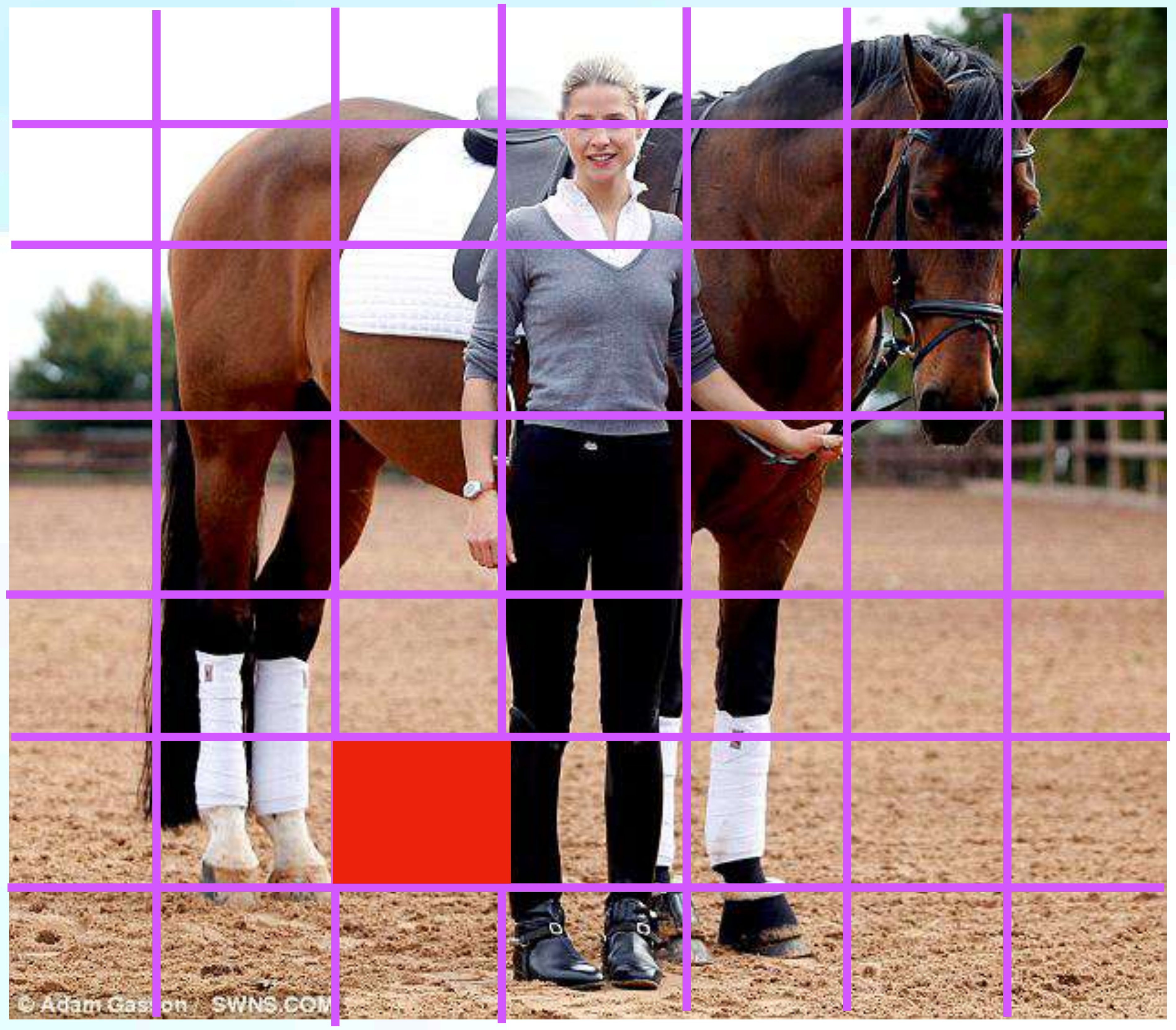


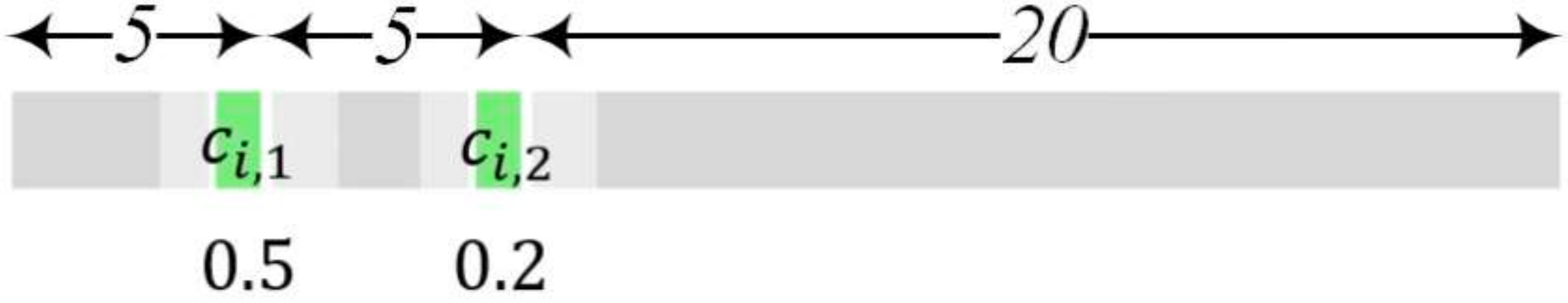
$$\hat{p}_{i,14=\text{person}} = 1.0$$

Loss for no-object cells

- Only calculates the objectness confidence score loss

$$L_{i,no\_obj} = (0.5 - \underbrace{0.0}_{}\!\!{}^{\hat{c}_{i,1}})^2 + (0.2 - 0.0)^2 = 0.29$$





Total Loss

$$\boldsymbol{L} = \lambda_{coord} \times \sum_{i=1}^{S^2} 1_i^{obj} \times \left(\begin{array}{c} (\Delta x_i^* - \Delta \hat{x}_i)^2 + (\Delta y_i^* - \Delta \hat{y}_i)^2 + \\ \left(\sqrt{\Delta w_i^*} - \sqrt{\Delta \widehat{w}_i}\right)^2 + \left(\sqrt{\Delta h_i^*} - \sqrt{\Delta h_i}\right)^2 \end{array} \right)$$

$$+ \sum_{i=1}^{S^2} 1_i^{obj} \times (c_i^* - \hat{c}_i)^2 \; + \sum_{i=1}^{S^2} 1_i^{obj} \times \sum_{c=1}^{20} \left(p_{i,c} - \hat{p}_{i,c}\right)^2$$

$$+ \lambda_{no\_obj} \sum_{i=1}^{S^2} 1_i^{no\_obj} \times \sum_{j=1}^{B} \left(c_{i,j} - \hat{c}_{i,j}\right)^2$$

Total Loss

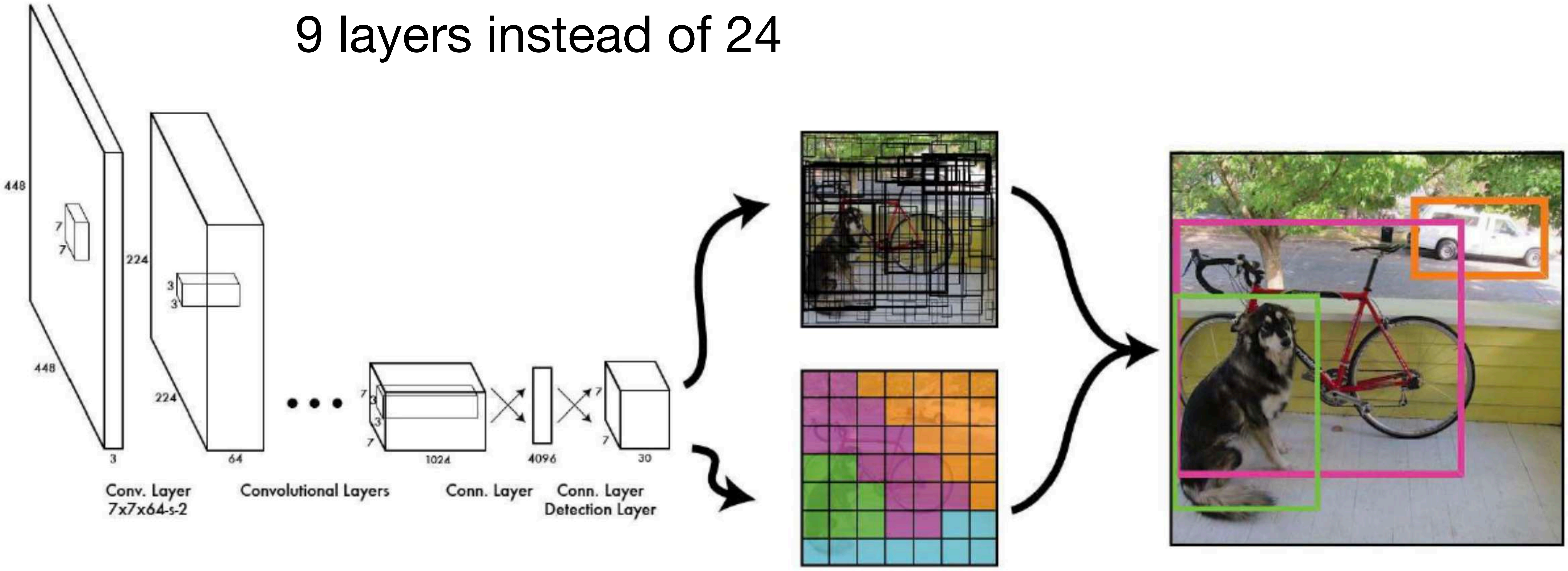
$$\boldsymbol{L} = \lambda_{coord} \times \sum_{i=1}^{S^2} 1_i^{obj} \times \left(\begin{array}{c} (\Delta x_i^* - \Delta \hat{x}_i)^2 + (\Delta y_i^* - \Delta \hat{y}_i)^2 + \\ \left(\sqrt{\Delta w_i^*} - \sqrt{\Delta \widehat{w}_i}\right)^2 + \left(\sqrt{\Delta h_i^*} - \sqrt{\Delta h_i}\right)^2 \end{array} \right)$$

$$+ \sum_{i=1}^{S^2} 1_i^{obj} \times (c_i^* - \hat{c}_i)^2 + \sum_{i=1}^{S^2} 1_i^{obj} \times \sum_{c=1}^{20} (p_{i,c} - \hat{p}_{i,c})^2$$

$$+ \lambda_{no\_obj} \sum_{i=1}^{S^2} 1_i^{no\_obj} \times \sum_{j=1}^{B} (c_{i,j} - \hat{c}_{i,j})^2$$

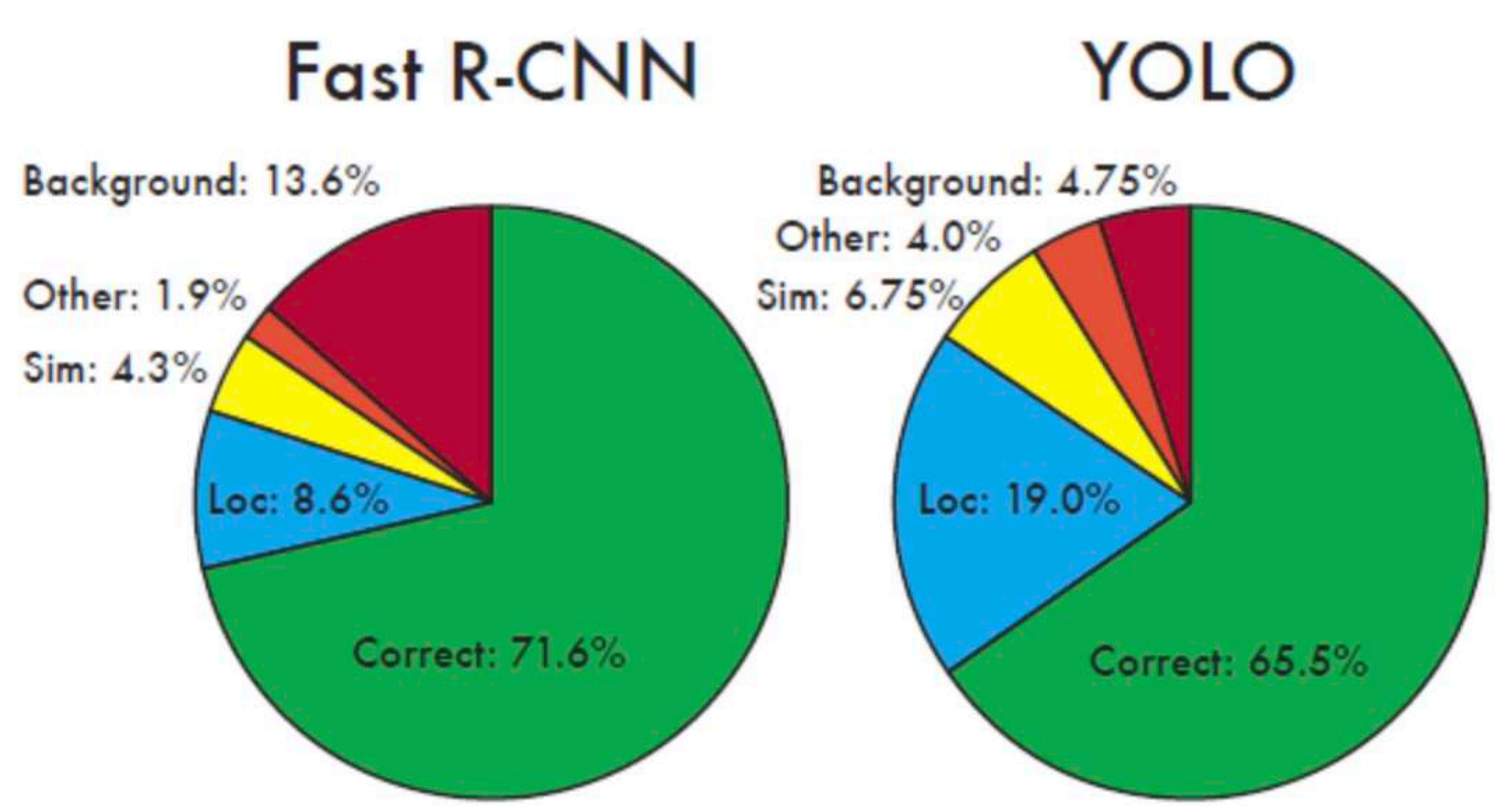
Fast YOLO

9 layers instead of 24

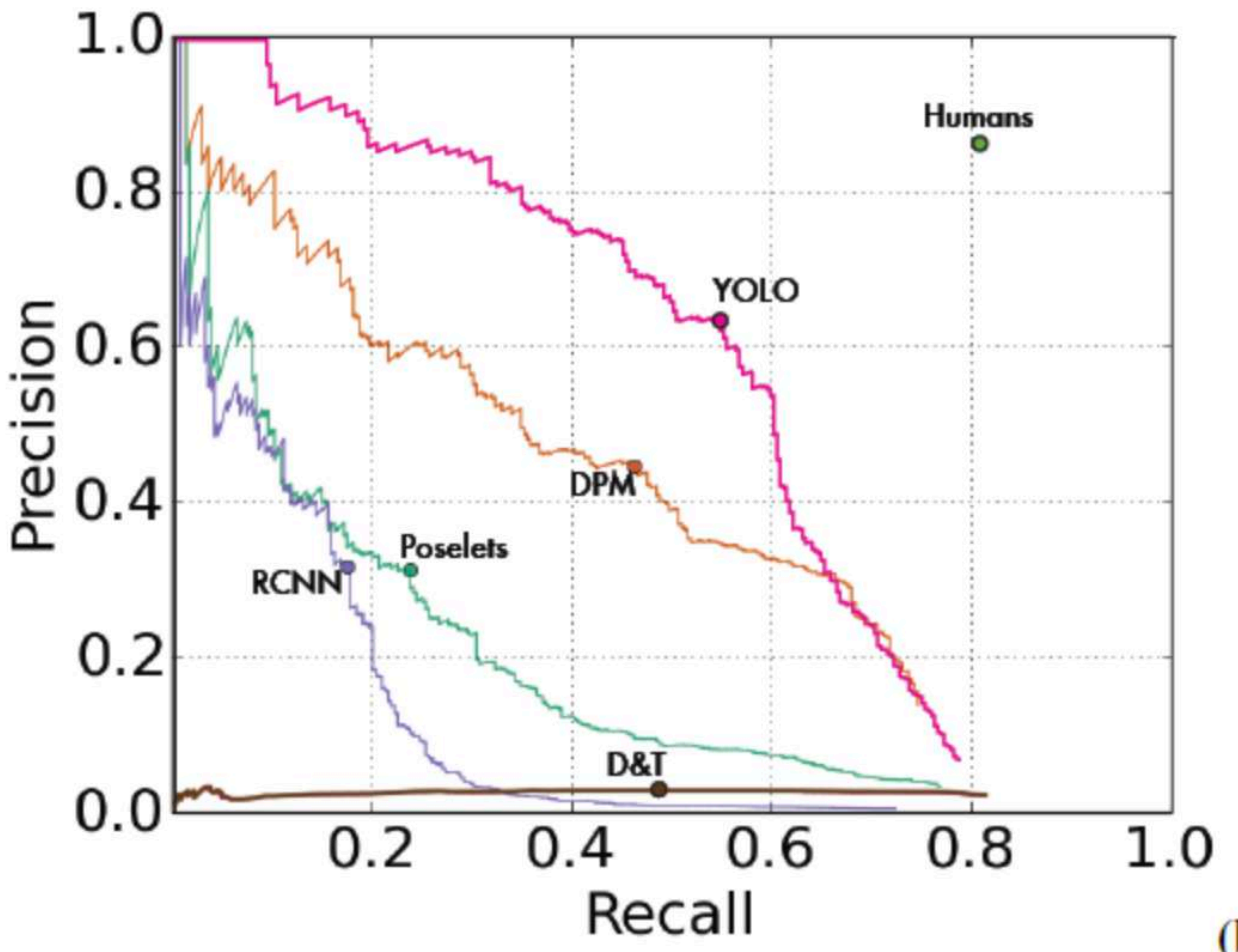


Performance

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21



Generalization Ability



(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

Limitations

- Maximum of 49 objects can be detected
- Difficulty in detecting small objects that appear in groups
- Poor localization

https://arxiv.org › cs

[1612.08242] YOLO9000: Better, Faster, Stronger - arXiv

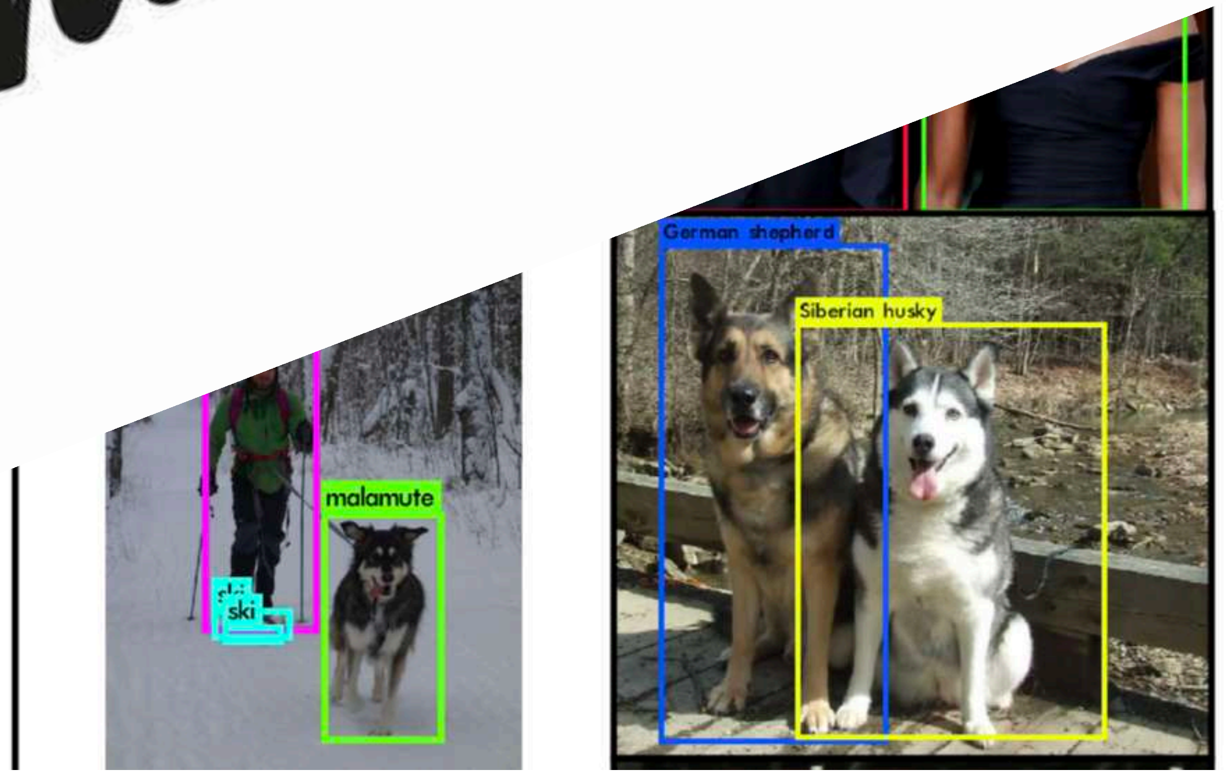
by J Redmon · 2016 · Cited by 14155 — The improved model, **YOLOv2**, is state-of-the-art on standard detection tasks like PASCAL VOC and COCO. At 67 FPS, **YOLOv2** gets 76.8 mAP o…

YOLO9000: Better, Faster, Stronger

Joseph Redmon*†, Ali Farhadi*†
University of Washington*, Allen Institute for AI†
http://pjreddie.com/yolo9000/

Abstract

We introduce YOLO9000, a state-of-the-art, real-ti
object detection system that can detect over 90
categories. First we propose various im
YOLO detection method, both no
work. The improved model, YO
standard detection tasks like PAS
ing a novel, multi-scale training
model can run at varying sizes, o
between speed and accuracy. At
76.8 mAP on VOC 2007. At 40 F
mAP, outperforming state-of-the-art m
CNN with ResNet and SSD while still r
faster. Finally we propose a method to
ject detection and classification. Using th
YOLO9000 simultaneously on the COCO
and the ImageNet classification dataset. Ou
allows YOLO9000 to predict detections for
that don't have labelled detection data. We ... ar
approach on the ImageNet detection task. YOL...000 gets
19.7 mAP on the ImageNet detection validation set despite
only having detection data for 44 of the 200 classes. On the
156 classes not in COCO, YOLO9000 gets 16.0 mAP. But



Thank You!