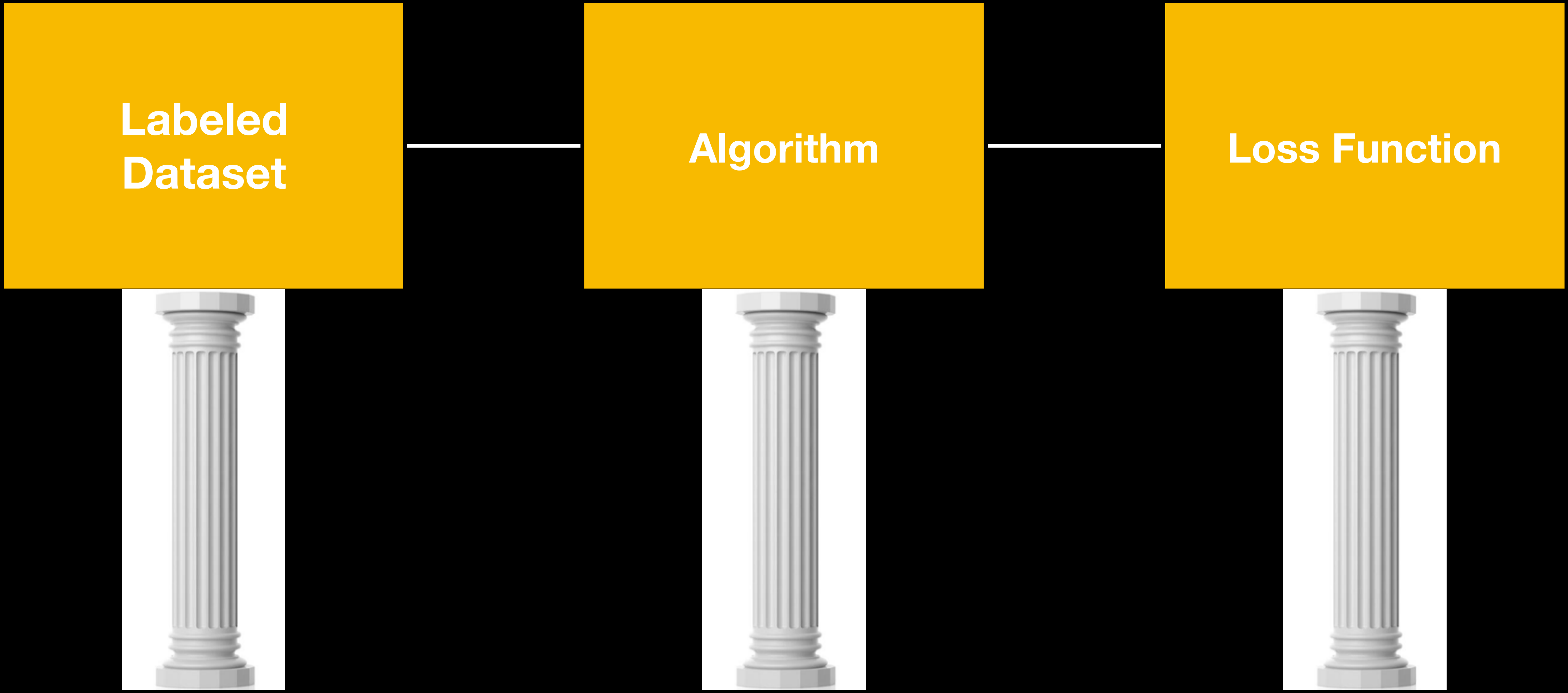
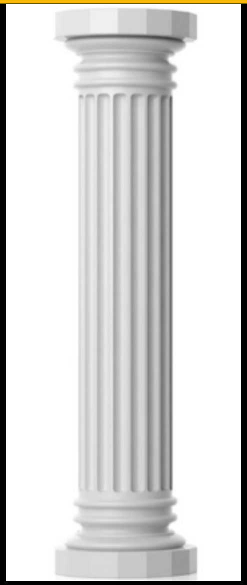
Supervised Learning: Object Detection

Earl Wong


Labeled Dataset
Algorithm
Loss Function

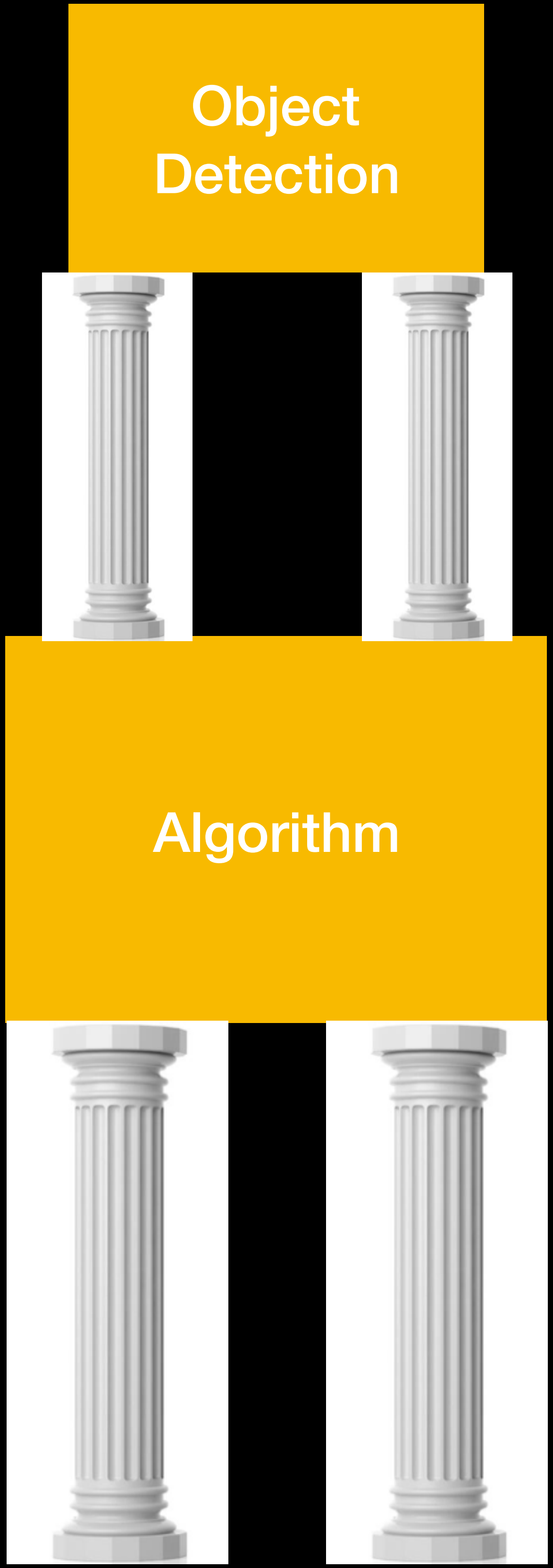
Labeled
Dataset



Labeled Dataset

Dataset	Pascal VOC 2012	ILSVRC 2013	MS COCO 2014
Classes	20	200	80
Images	11K	415K	200K
Objects	27K	400K	500K
Avg Resolution	469 x 387	482x415	Did not state (comparable in resolution)

Person: person
Animal: bird, cat, cow, dog, horse, sheep
Vehicle: plane, bicycle, boat, bus, car, motorbike, train
Indoor: bottle, chair, table, plant, sofa, tv/monitor

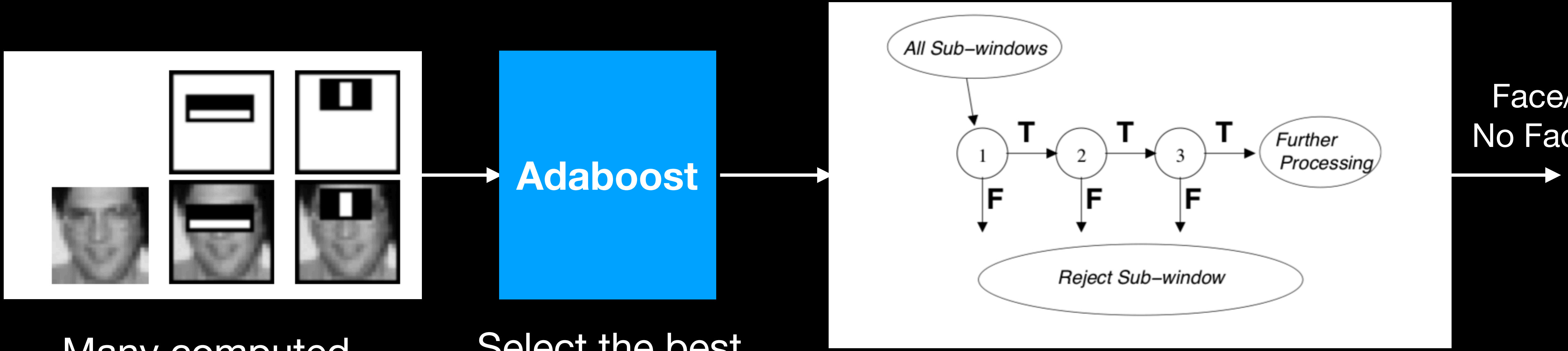

Object Detection
Algorithm

Historical

In the pre-AlexNet era, two well known object detection systems included:

- 1) Face detection (Viola and Jones) - Wavelet feature set
- 2) Human detection (Dalal and Triggs) - HOG feature set

Face Detection



Many computed features

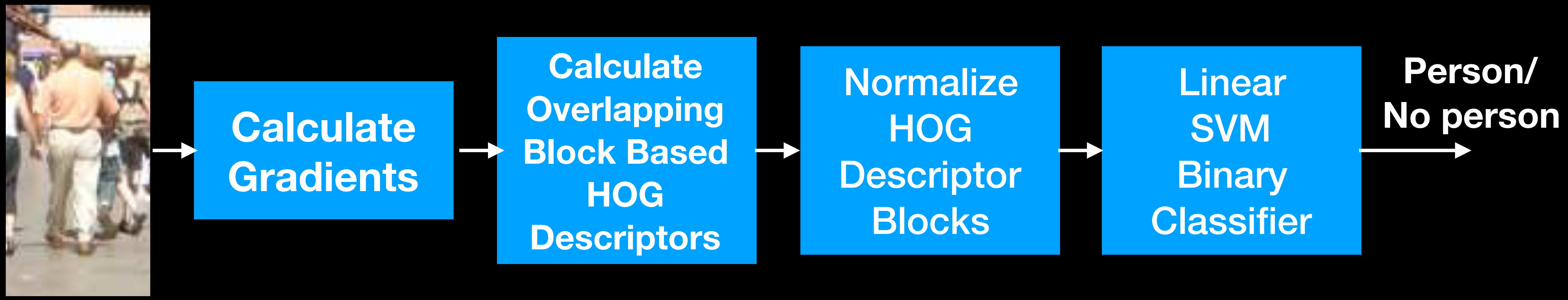
Select the best features for the initial classification

Continue to refine the result, by applying more “fine grain” classifiers

There were 3 main contributions that led to the deployment of this system:

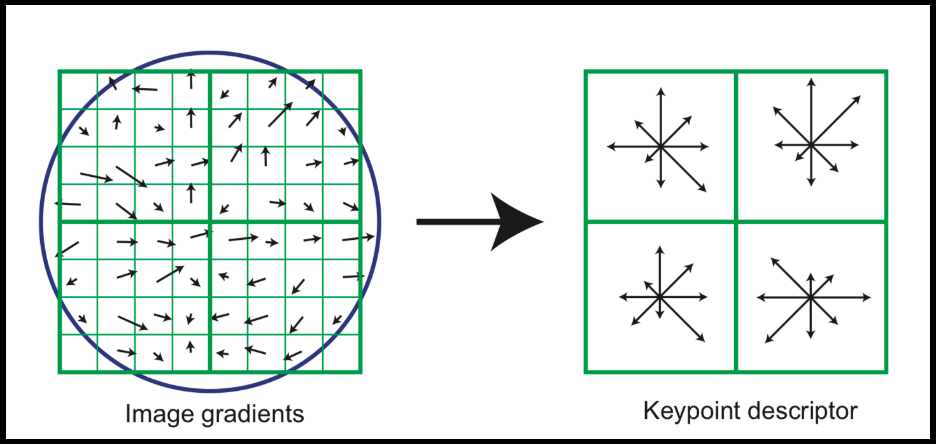
1) +The use of 2D Haar basis functions and their rapid computation using integral images.
2) +The use of Adaboost to select the best features.
3) +The refinement of the initial results, by performing additional boosting (via cascade) on the more challenging features.

Human Detection


Calculate Gradients
Calculate Overlapping Block Based HOG Descriptors
Normalize HOG Descriptor Blocks
Linear SVM Binary Classifier
Person/ No person

1) First, the image gradients were computed in order to generate HOG feature descriptors for a given window size = cell / block.

2) The HOG descriptors were similar to the SIFT descriptor used for image matching.



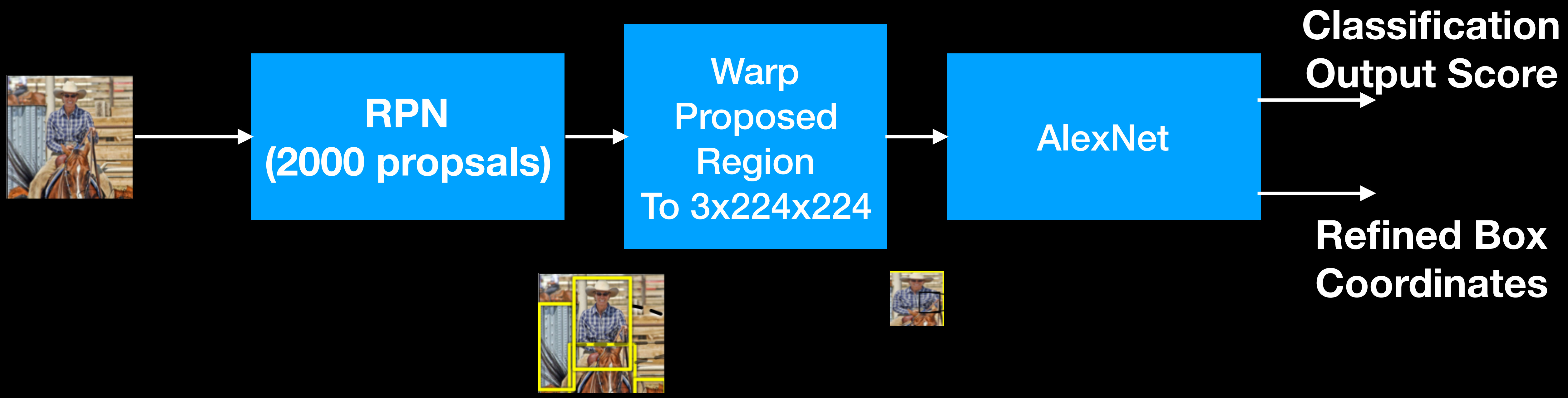
3) Next, the descriptors were normalized and then fed into the classifier. (This was a crucial step for successful classification.)

Note: The authors developed a new test set for this task, because the initial test set (see image above) was too easy.

Deep Network Era

- R-CNN (2 stage detector)
- Fast R-CNN (2 stage detector)
- Faster R-CNN (2 stage detector) {extension to pose estimation and segmentation using Mask R-CNN}
- YOLO (1 stage detector)
- SSD (1 stage detector)

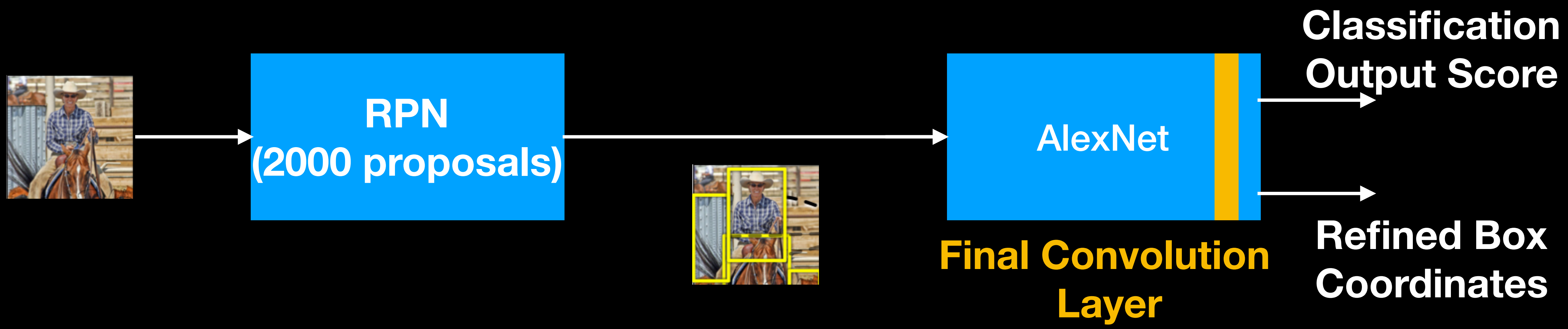
R-CNN
Idea1: Use AlexNet as a Classifier and a Regressor



1) The selective search algorithm (RPN) proposed 2000 candidate object regions.
2) These regions were cropped and warped into 3x224x224 tensors for input into AlexNet.
3) AlexNet was (pre-trained on ImageNet) and fine tuned on Pascal VOC. In addition, the authors added a pre-trained bounding box regressor unit to the original AlexNet.
4) Using a fixed threshold, only classification results exceeding the threshold were taken.
5) Non-maximal suppression (NMS) was then applied to eliminate overlapping proposal regions.

Fast R-CNN

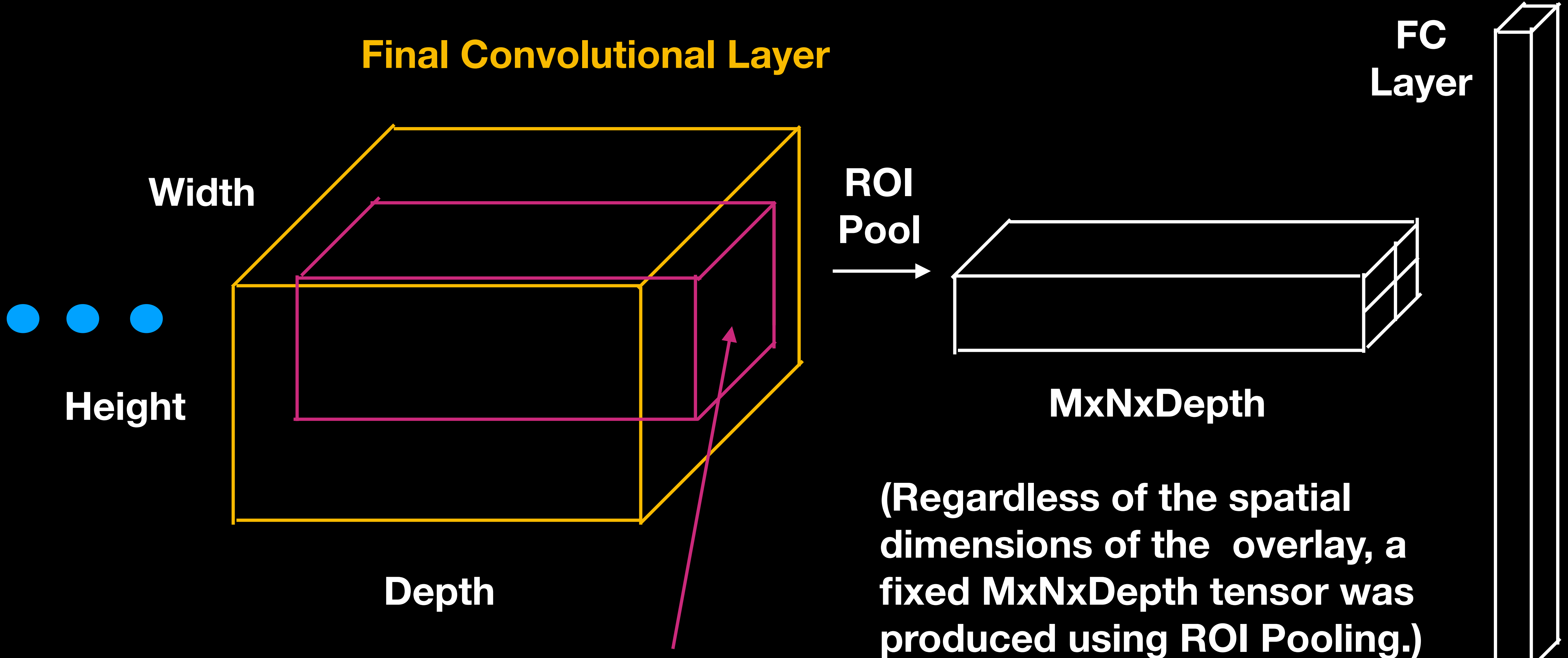
Idea2: Map All Proposed Regions to a Fixed Interface Tensor



1) Warping 2000 region proposals for sequential processing by AlexNet was 2000x too slow.
2) Instead, the region proposals were now overlaid onto the final convolutional layer of AlexNet.
3) An additional remapping (ROI Pooling) was then performed, producing a **fixed size** MxNxD tensor. (See next slide.)
4) The MxNxD tensor was then connected to the fully connected layer in the usual manner.

Fast R-CNN

Idea2: Map All Proposed Regions to a Fixed Interface Tensor

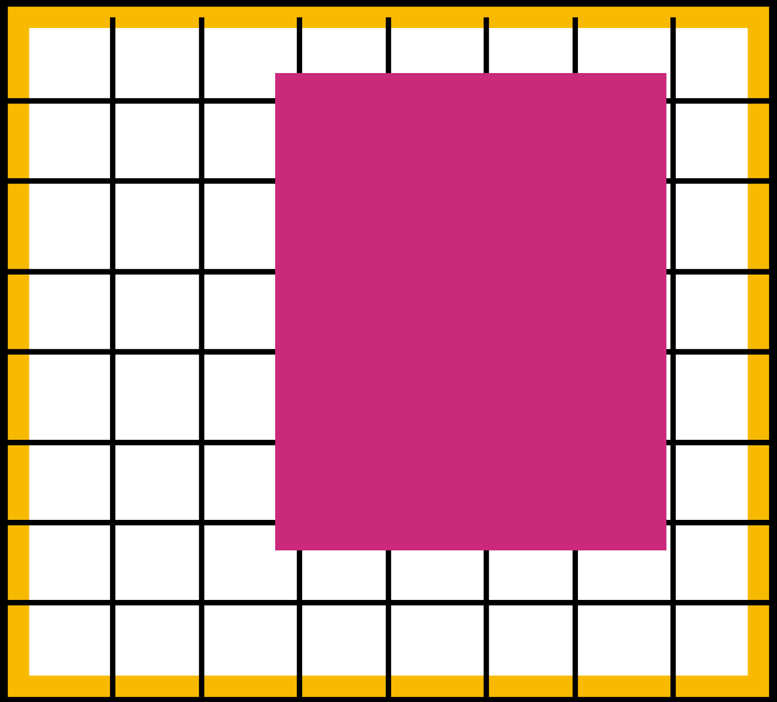


(Regardless of the spatial dimensions of the overlay, a fixed MxNxDepth tensor was produced using ROI Pooling.)

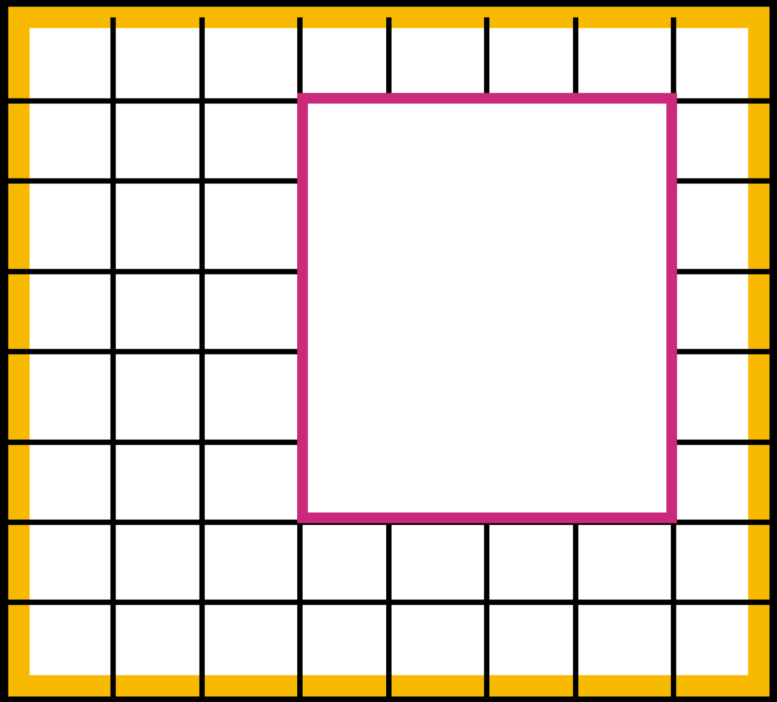
A candidate region proposal is overlaid onto the final CNN layer.

Fast R-CNN

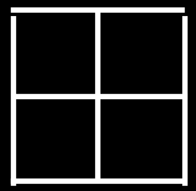
Idea2: Map All Proposed Regions to a Fixed Interface Tensor



Region proposal is overlaid onto the final CNN layer



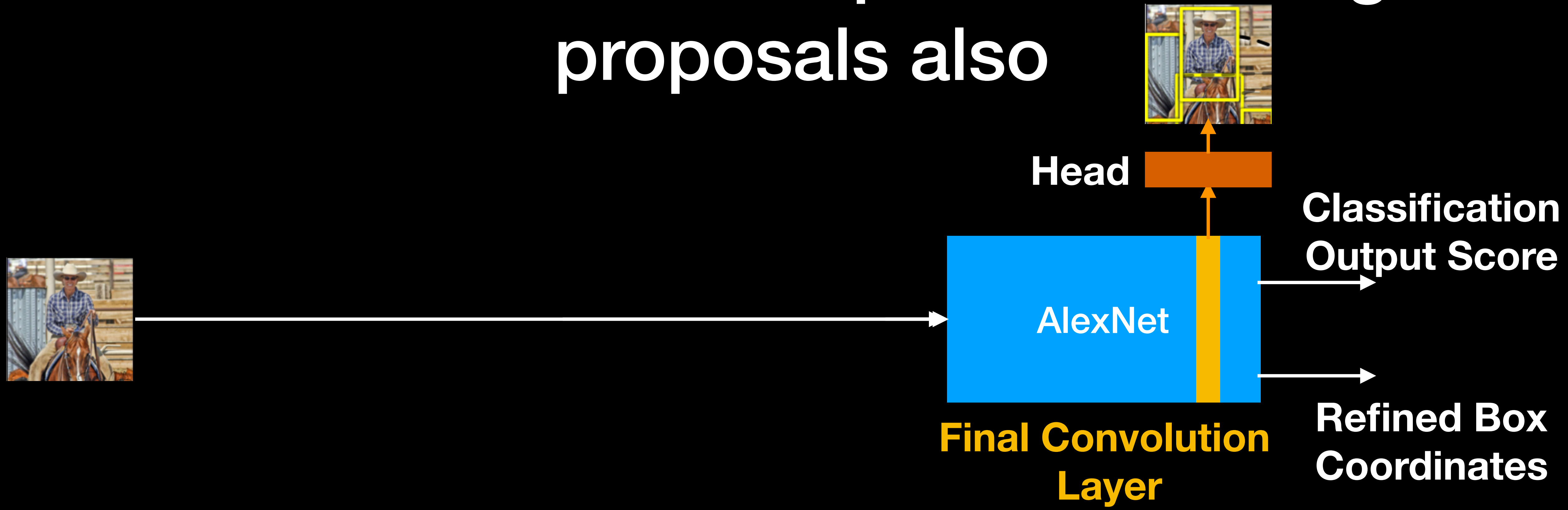
Coarse snapping to the grids occurs



If the desired output is 2x2:
2x2, 2x2, 2x3 and 2x3 ROI Pooling is then applied

In this example, the proposed region is mapped to a fixed spatial dimension output of (M = 2) x (N = 2) using ROI Pooling.

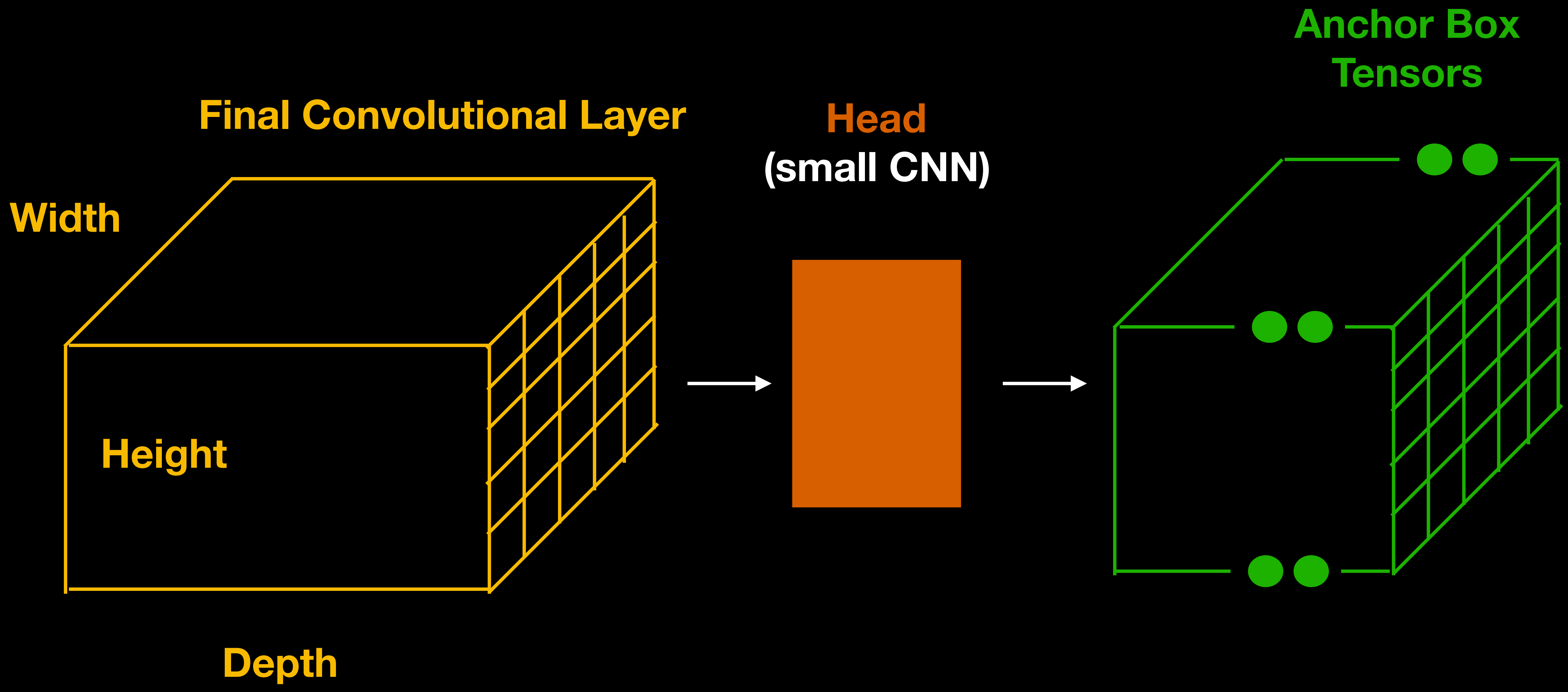
Faster R-CNN
Idea3: Make the CNN produce the region proposals also



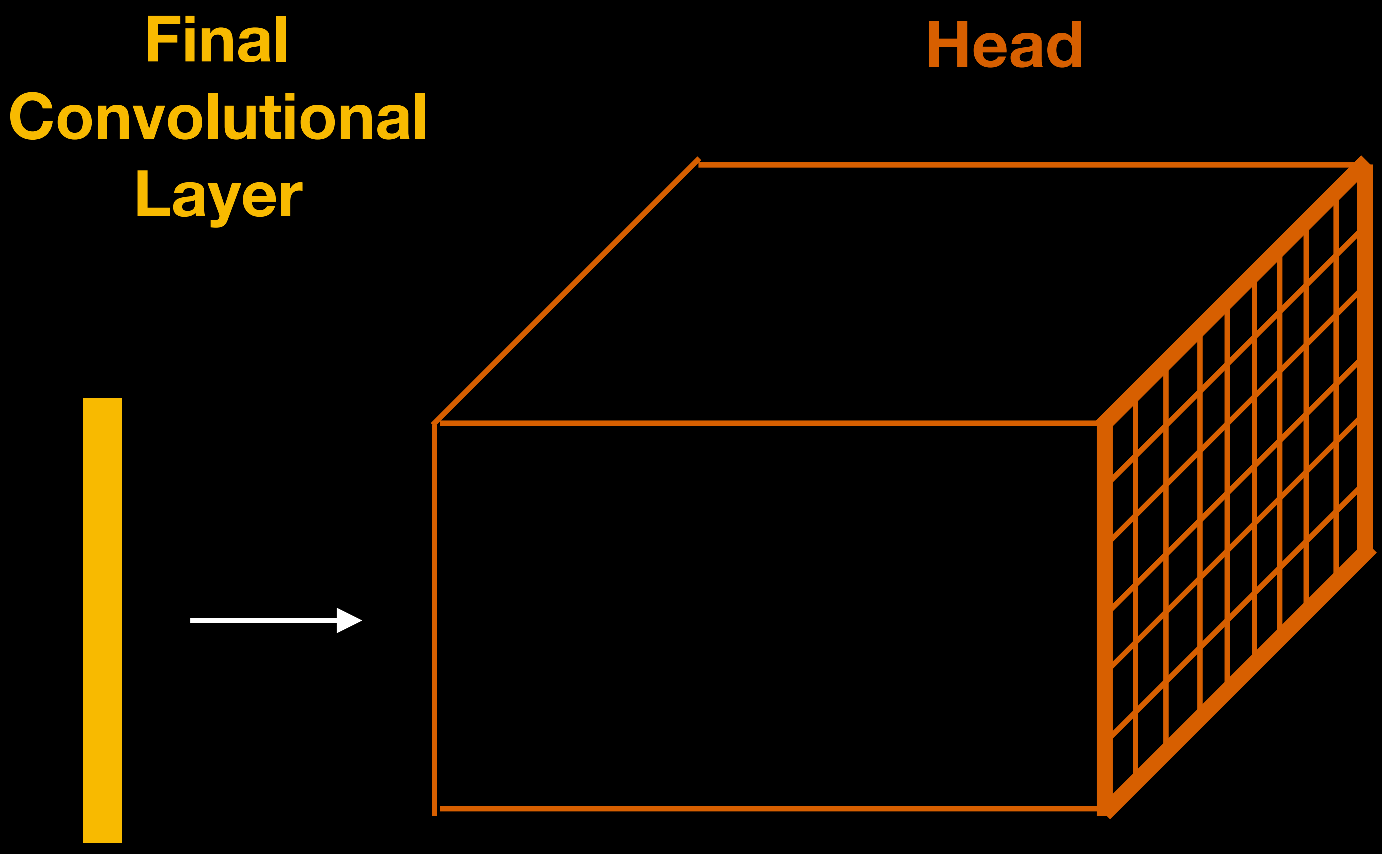
1) Using the previous setup, the speed bottleneck was now associated with the region proposal algorithm.
2) To remedy this, a new output **head** (small CNN) was added to AlexNet to produce the region proposals.
3) This head utilized the notion of anchor boxes, for fast proposal outputs.
4) As a result, the subsequent loss function was increased from 2 terms to 4 terms: 2 terms were associated with the region proposal (object/no object classification and initial bounding box estimate) and 2 terms were associated with the output (class object and refined bounding box estimate).
5) This made training fairly tricky and heuristic. However, it did work!

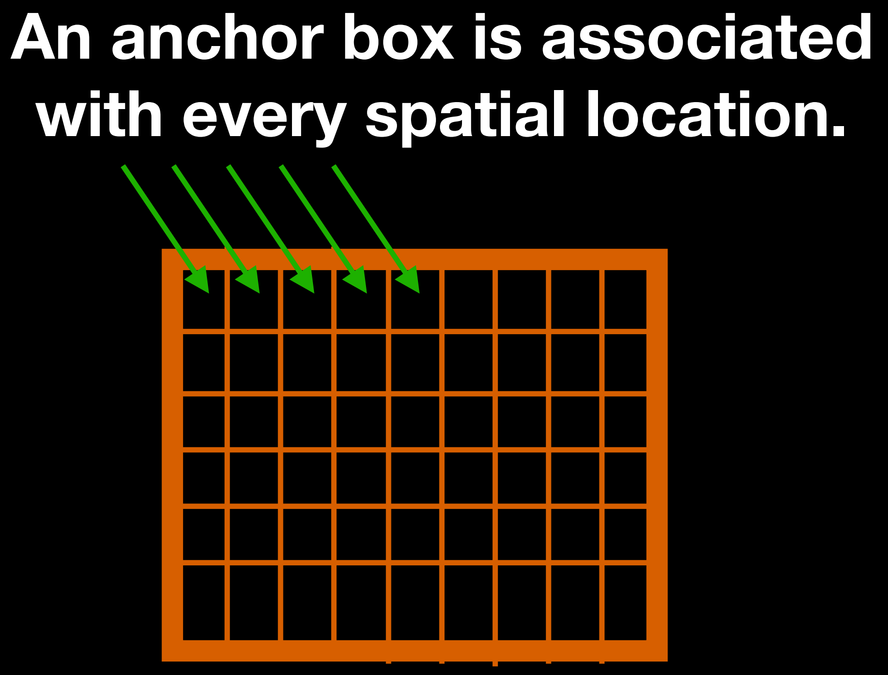
Faster R-CNN
Idea3: Make the CNN produce the region proposals also

A head (small CNN) is added between the final convolutional layer of the backbone network and the desired anchor box tensors.



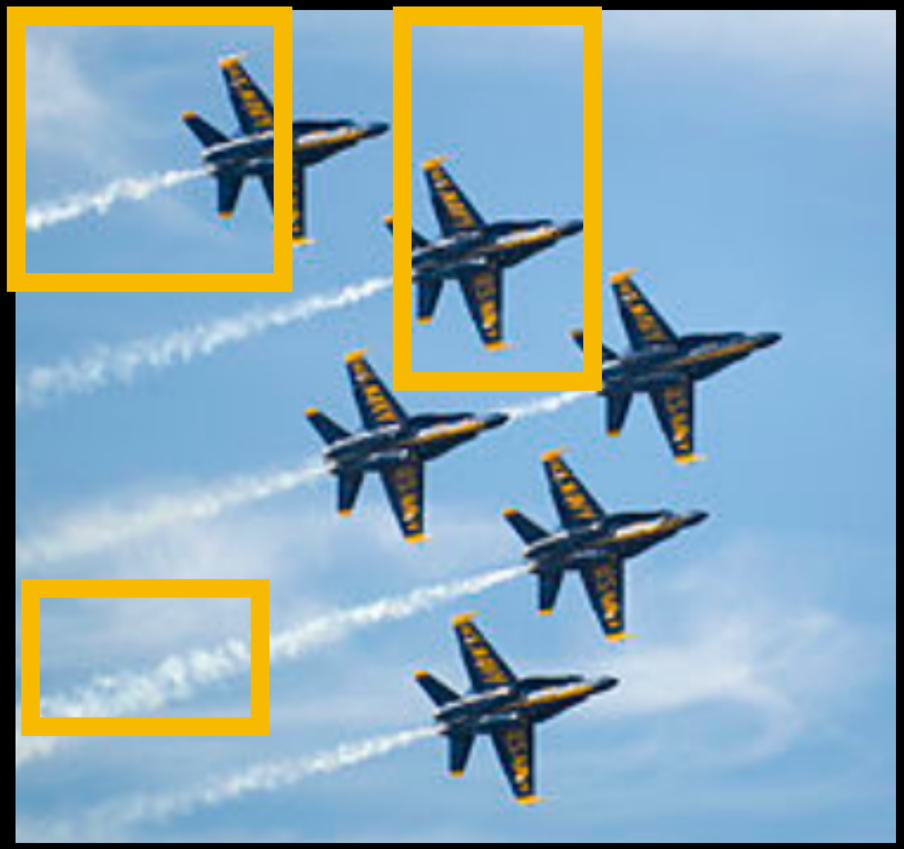
Faster R-CNN
Idea3: Make the CNN produce the region proposals also





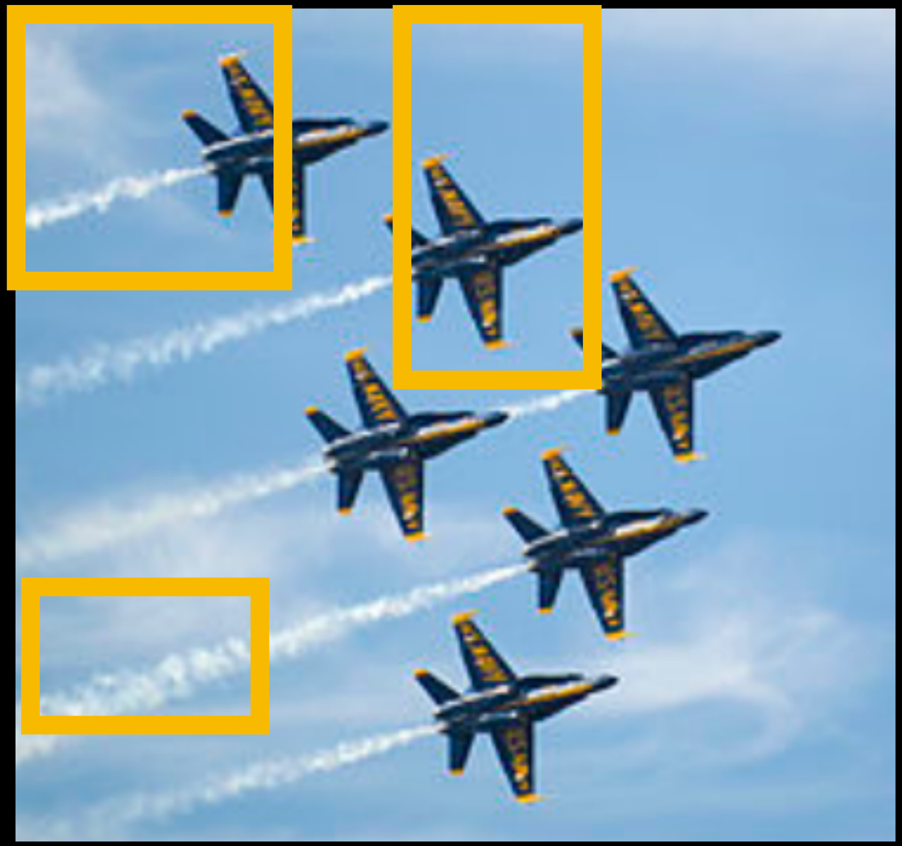
1) Each spatial location in the output head was associated with an anchor box tensor.
2) Each anchor box tensor contained a binary classification object / no object result and an initial bounding box estimate.
3) This information was then used in the usual fashion, per Fast R-CNN.

The Motivation For Anchor Boxes



1) Classical computer vision utilized sliding windows (of various sizes and shapes) to detect objects in the input image.
2) In deep learning, the natural extension would be to perform a similar analysis, using the feature maps of a CNN.
3) However, sliding windows are computationally intensive.
4) Instead, anchor boxes are used as a substitute.

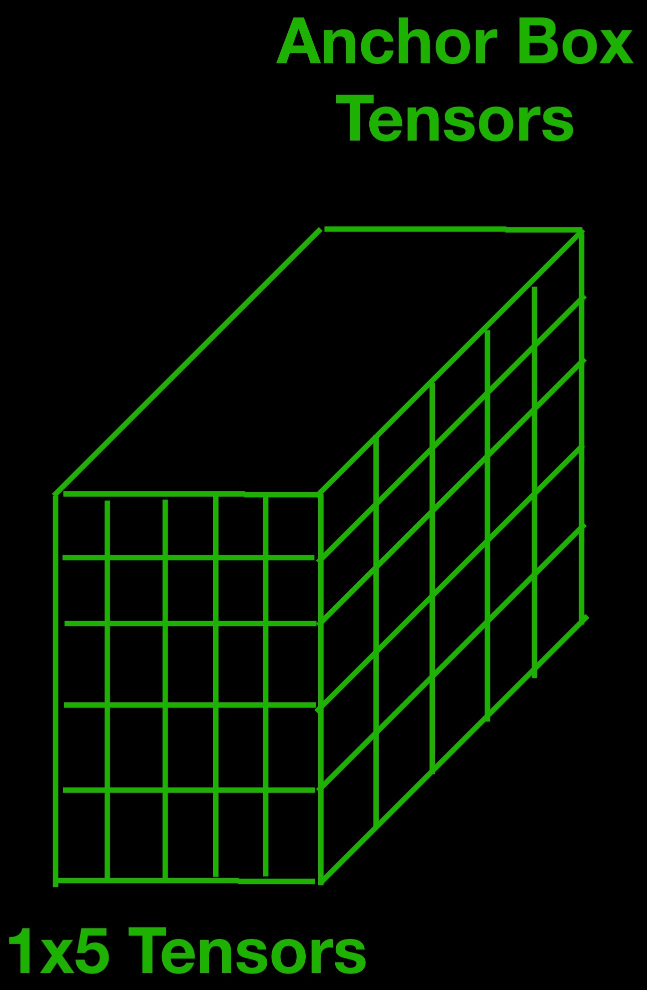
How Do Anchor Boxes Work?

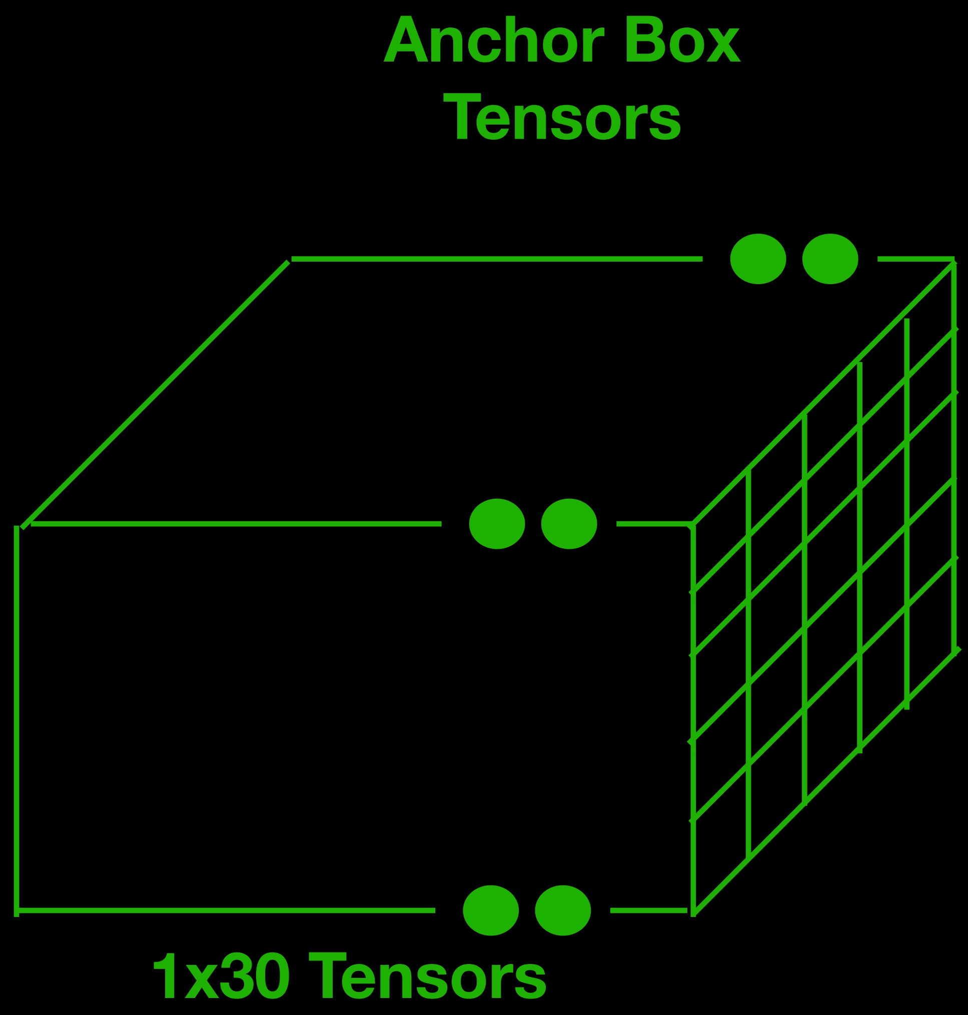


1) At every spatial location, an anchor box tensor is generated.
2) This N-dim tensor contains classification and bounding box coordinate information.
3) For example, the tensor might consist of a binary classification result and four bounding box coordinates, resulting in a 1x5 tensor.
4) Or, the tensor might consist of a binary classifier result and eight bounding box coordinates (for 2 aspect ratios), resulting in a 1x9 tensor, etc.
5) Through extensive training, the network learns what the outputs should be.

Note: Candidate objects can / will typically span multiple spatial locations. The best bounding box coordinate estimates will output a bounding box estimate that encloses the entire object. To help, the anchor box dimensions are pre-empted with an aspect ratio / initial dimensions.

Faster R-CNN <—> YOLO

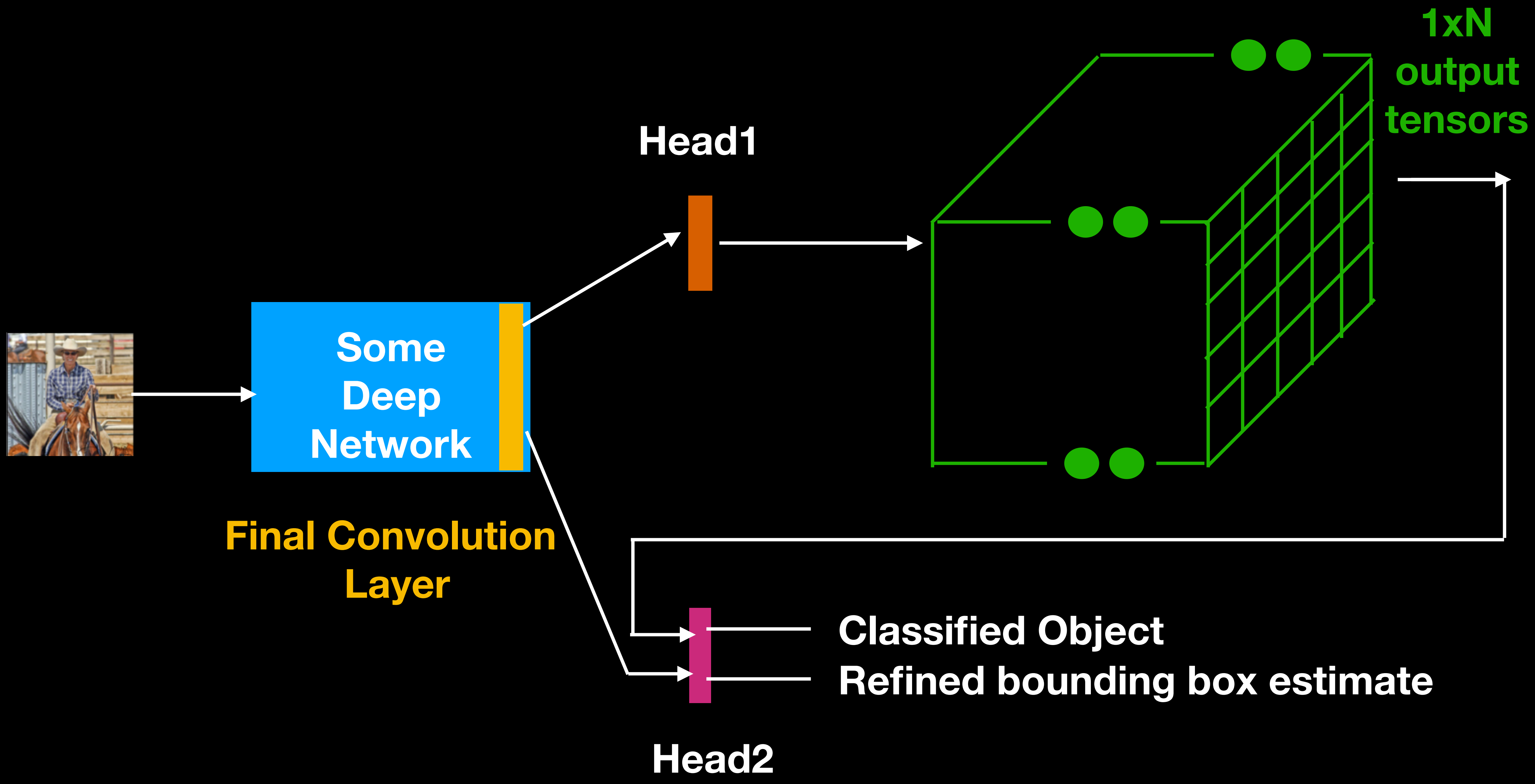




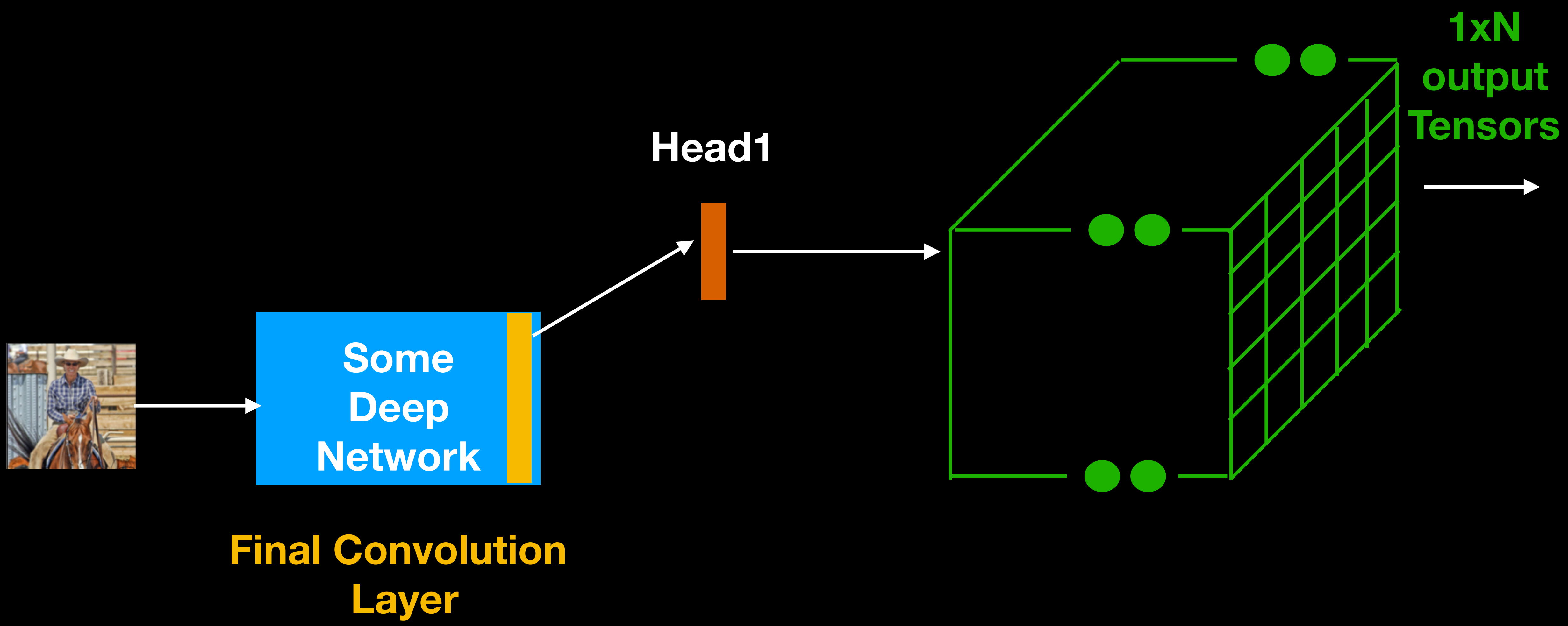
Faster R-CNN
Output tensor at every spatial location = 1x5 {Object/No Object, 4 Bounding Box Coordinate Estimates}

YOLO
Output tensor at every spatial location = 1x30 {Probability Estimate for 20 Objects, 4 Bounding Box Coordinate Estimates for two different aspect ratios, 2 Confidence Estimates}

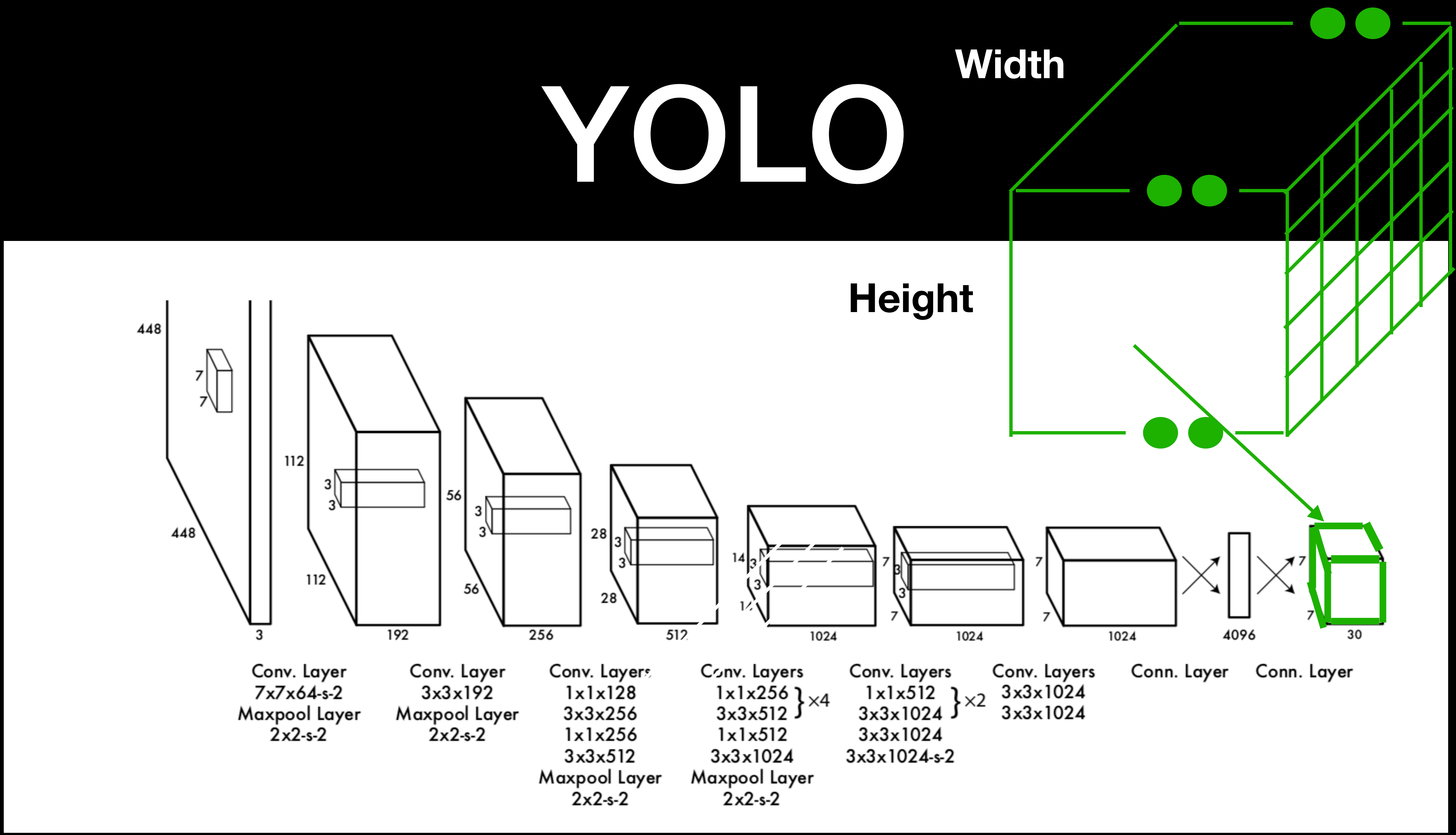
2 Stage Detector versus 1 Stage Detector



2 Stage Detector versus 1 Stage Detector

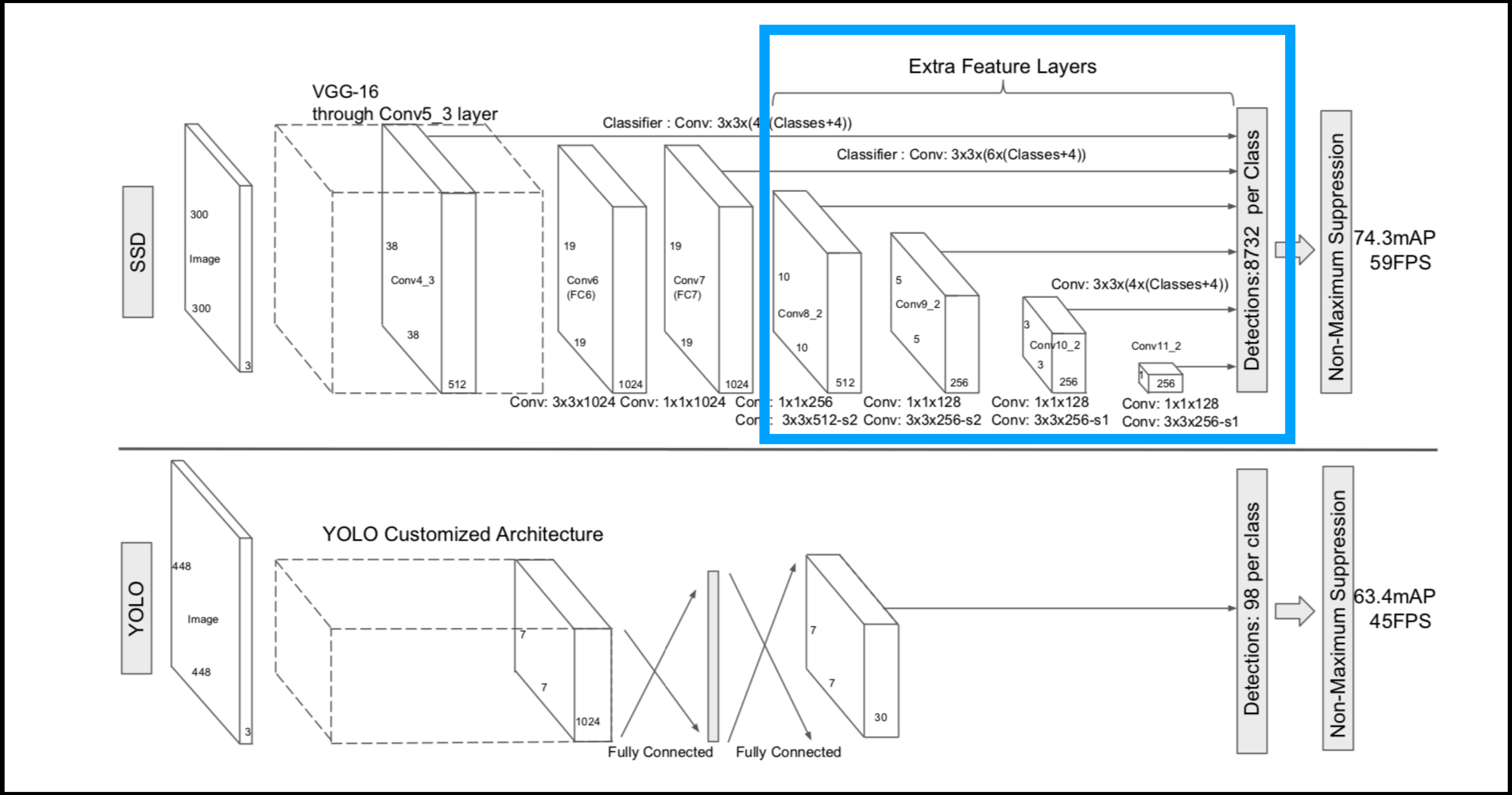


YOLO


Width
Height
448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

1) YOLO was a single stage object detector that utilized anchor boxes.
2) YOLO was highly regarded for its real time object detection speed.
3) The main weakness with YOLO was its inability to detect small objects (such as a small flock of birds).
4) This was because multi-resolution capabilities were not (initially) architected into YOLO.

SSD

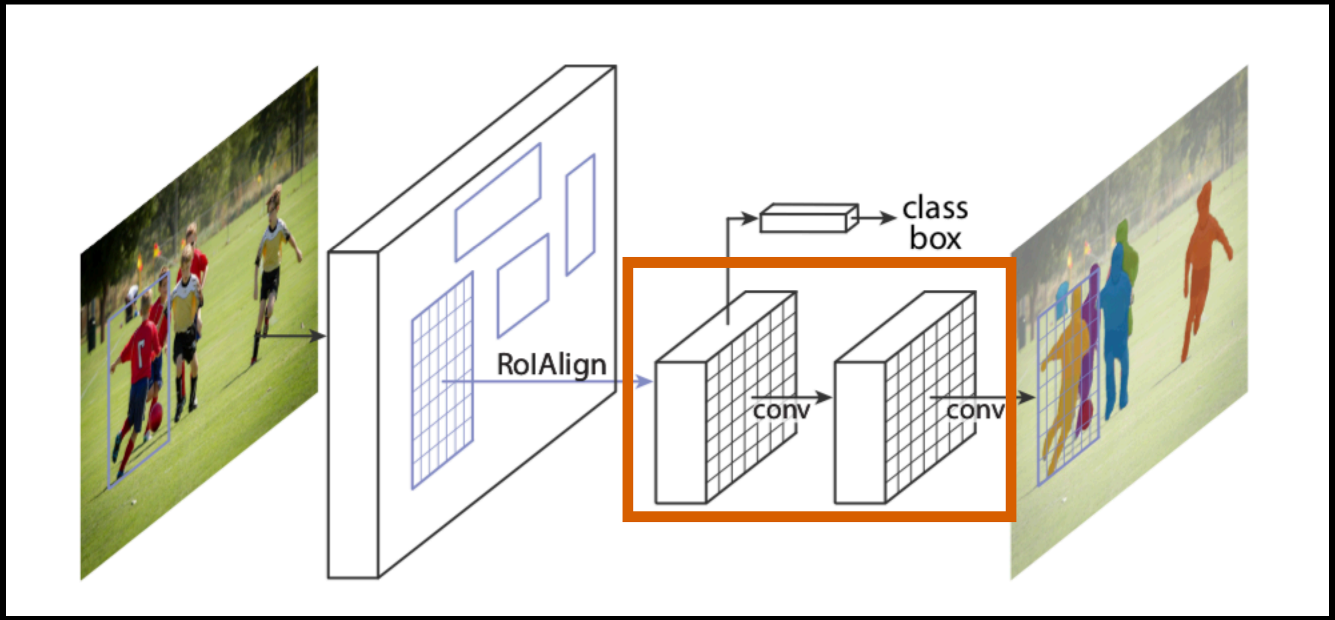


1) Like YOLO, SSD was a single stage detector.
2) However, **multi-resolution capabilities** were architected into SSD.
3) As a result, SSD significantly outperformed YOLO, when it came to detecting small objects.
4) SSD was slower than YOLO though.

The Best of All Worlds

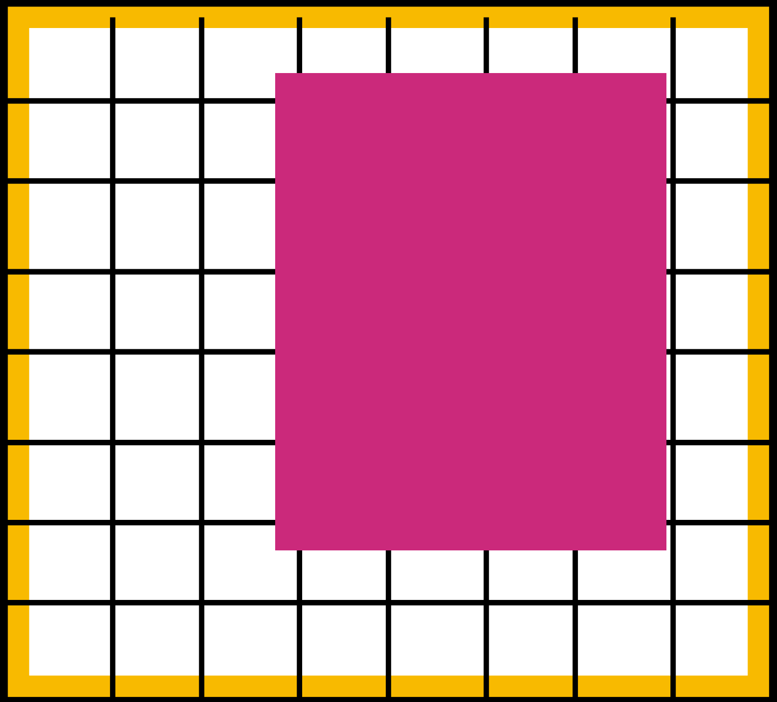
- By replacing sliding windows with anchor boxes, the speed of the object detector improved.
- By adding classical computer vision multi-resolution capabilities, the accuracy of the object improved.
- By permitting the refinement of the initial bounding box estimates, the accuracy and the localization of the object detector improved. (i.e. 2 stage detector versus 1 stage detector)
- By replacing the AlexNet backbone with any higher performing backbone (like ResNet), the accuracy and the localization of the object detector improved for R-CNN.

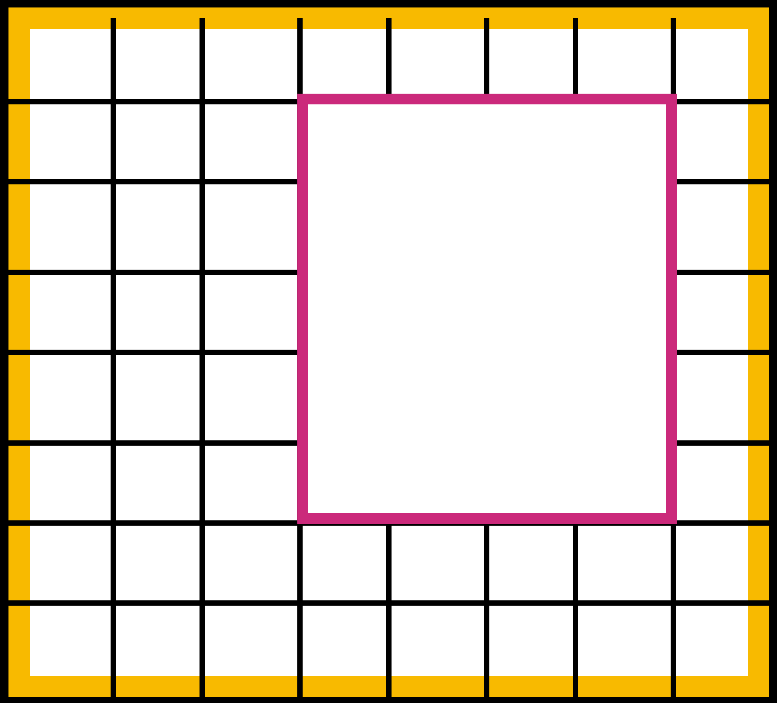
Mask R-CNN

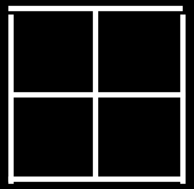


1) Additional pose estimation and instance segmentation results were produced from Faster R-CNN by adding additional output heads.
2) For instance segmentation, a reduced spatial resolution segmentation map was generated.
3) For pose estimation, a 17 channel output map was generated, where each channel contained detections for a specific pose joint.
4) In addition, in order to obtain more accurate results, ROI Align replaced ROI Pooling.

ROI Pool versus ROI Align

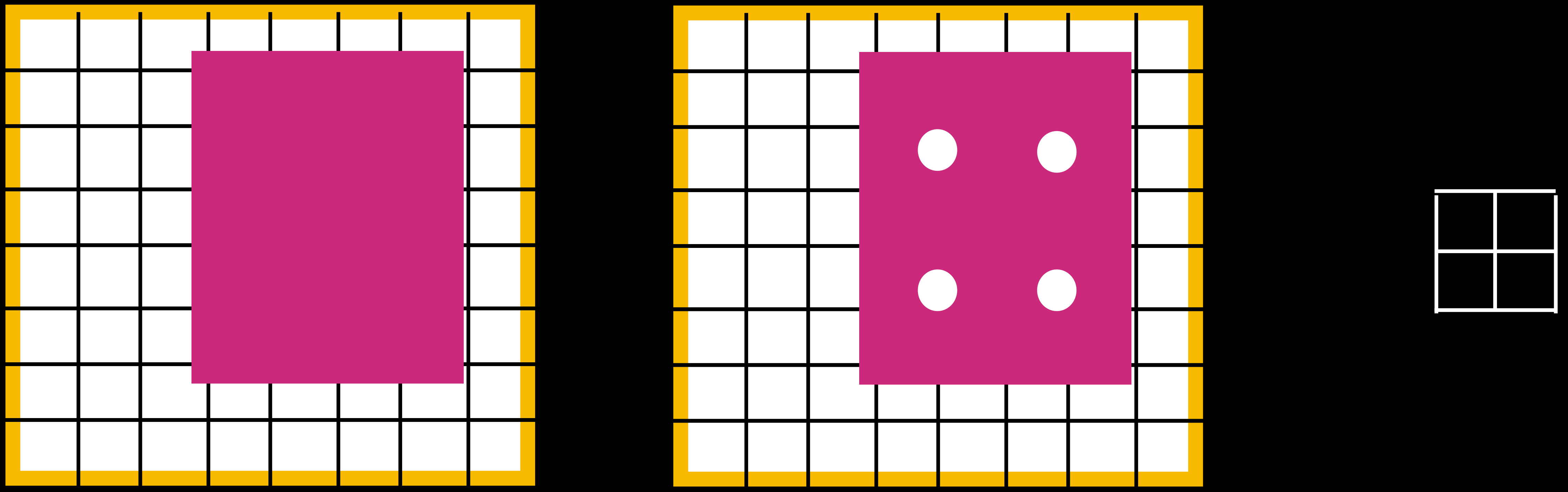

Region
proposal
is overlaid
onto the final
CNN layer


Coarse
snapping
to the
grids
occurs


If the desired output is
2x2:
2x2, 2x2, 2x3 and 2x3
ROI Pooling
is then applied

In this example, the proposed region is mapped to a fixed spatial dimension of (M = 2) x (N = 2) using ROI Pooling

ROI Pool versus ROI Align



Region proposal is overlaid onto the final CNN layer

Interpolation is then used to generate the output

In this example, the proposed region is mapped to a fixed spatial dimension of 2 x 2 using bilinear interpolation.

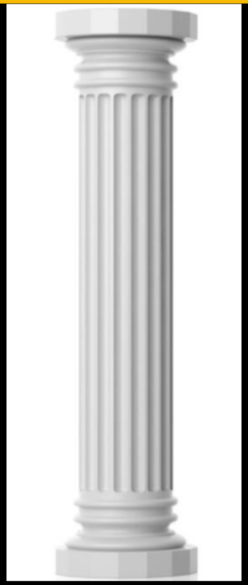
Algorithm (Architecture)

- R-CNN, Fast R-CNN, Faster R-CNN, Mask-R-CNN - Best Deep NN backbone from image classification + multiple output heads attached to the backbone + anchor boxes + multi-resolution information.

- YOLO - “Bare bones” vanilla CNN architecture + anchor boxes

- SGD - Vanilla CNN architecture + multi-resolution information aggregation from various CNN layers

Loss Function



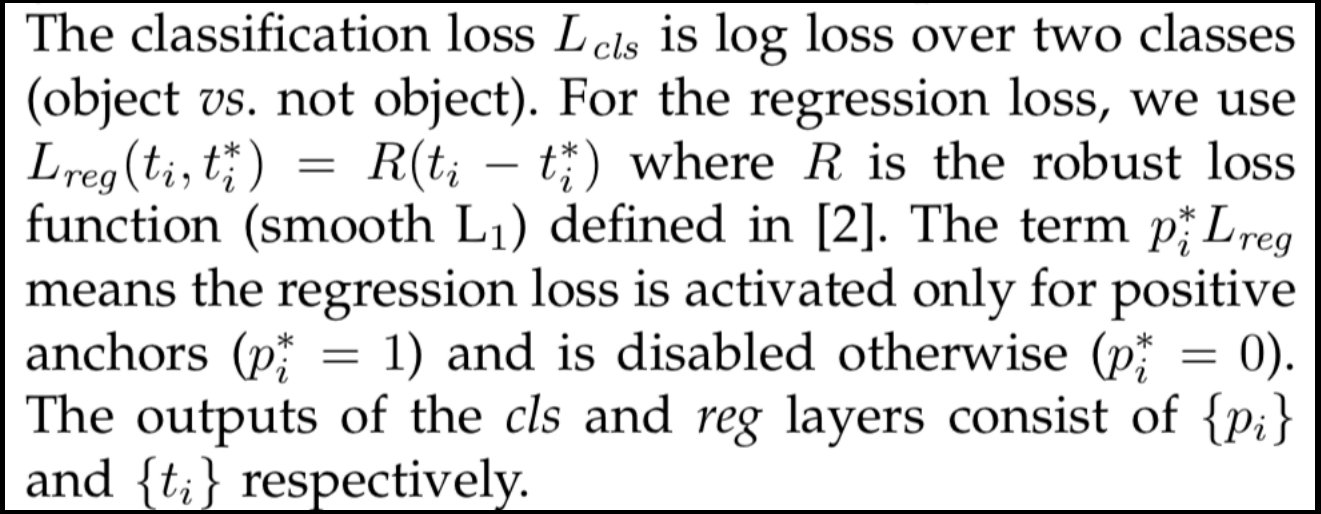
Loss Function

- Since the output consisted of two disjoint outputs (object classification and bounding box coordinate estimation), a multi-task loss function was used.
- For single stage detectors, the loss function contained 2 terms - one for predicting the object class and one for estimating the bounding box coordinates.
- The class prediction term was a cross entropy loss while the bounding box coordinate estimation was a regression loss.
- For the two stage detector Faster R-CNN, there were now four terms, since two additional terms were needed for the region proposal network.

Loss Function

Faster R-CNN

$$L(\{p_i\},\{t_i\}) = \boxed{\frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*)} + \lambda\boxed{\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*)}$$

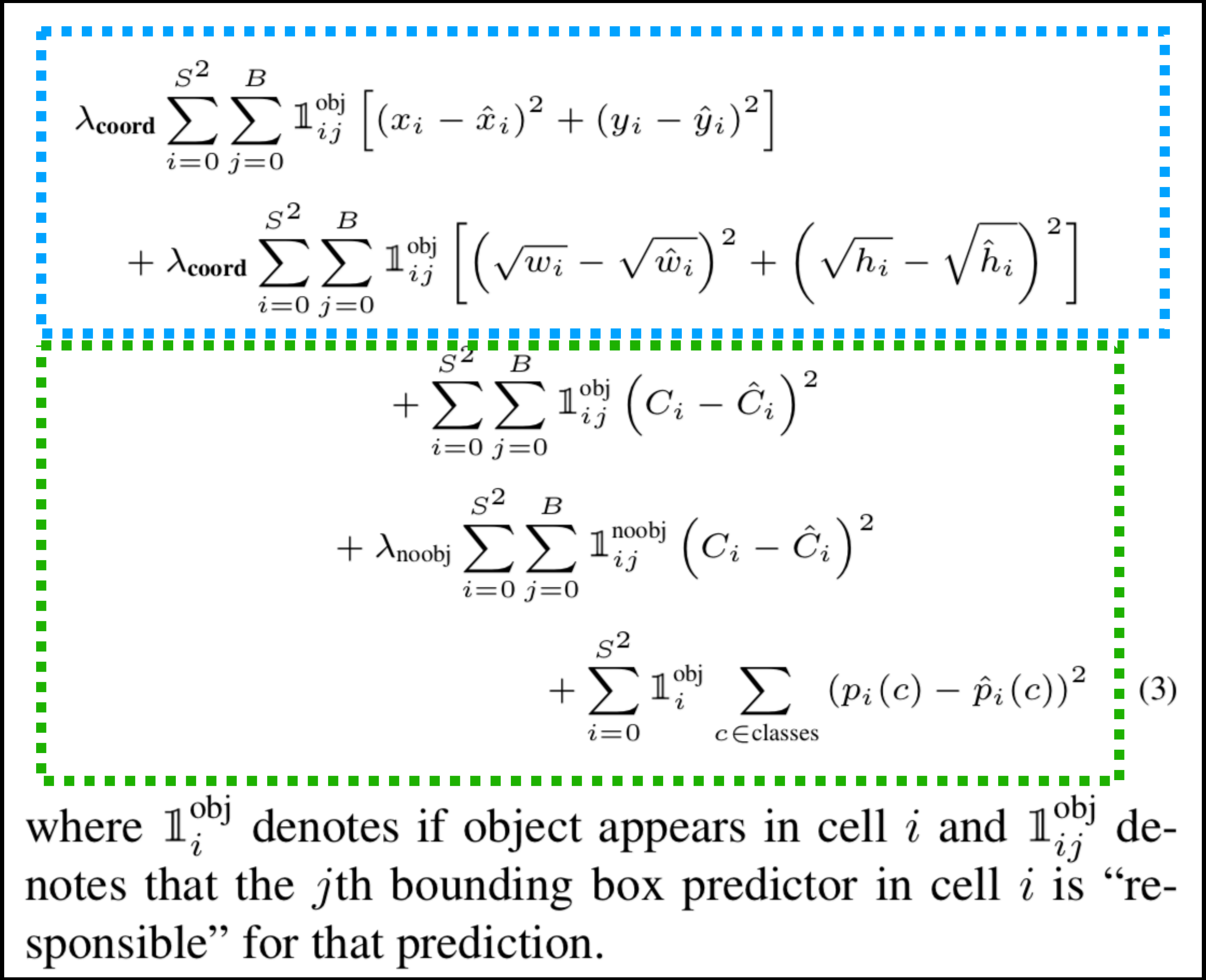


The classification loss L_{cls} is log loss over two classes (object *vs.* not object). For the regression loss, we use $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where R is the robust loss function (smooth L_1) defined in [2]. The term $p_i^* L_{reg}$ means the regression loss is activated only for positive anchors ($p_i^* = 1$) and is disabled otherwise ($p_i^* = 0$). The outputs of the *cls* and *reg* layers consist of $\{p_i\}$ and $\{t_i\}$ respectively.

Loss = rpnHeadClassLoss + lambda1 * rpnHeadRegressionLoss +
lambda2 * ouputClassLoss + lambda2 * ouputRegressionLoss

Loss Function

YOLO



$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \tag{3}$$

where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

Loss Function

SSD

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

tive [7,8] but is extended to handle multiple object categories. Let $x_{ij}^p = \{1, 0\}$ be an indicator for matching the i-th default box to the j-th ground truth box of category p.

mAP Evaluation Metric for Object Detection

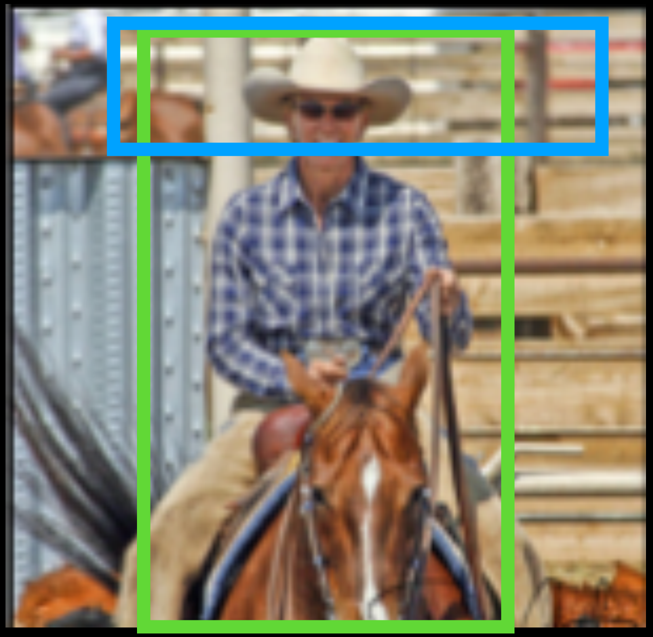
- mAP = mean of AP
- mAP = Given 20 object categories: 1) First, compute the AP for each category. 2) Second, average the AP results.
- What is AP?
- AP is computed as the area under the Precision vs. Recall curve.

mAP Evaluation Metric for Object Detection

- How do you compute the Precision versus Recall curve for an object category?

- First, perform NMS on the bounding box results produced by your algorithm.

- Second, you need the notion of whether an object has been detected or not. The metric for this is IoU - intersection over union.



IoU =

area where green and blue boxes overlap

— -

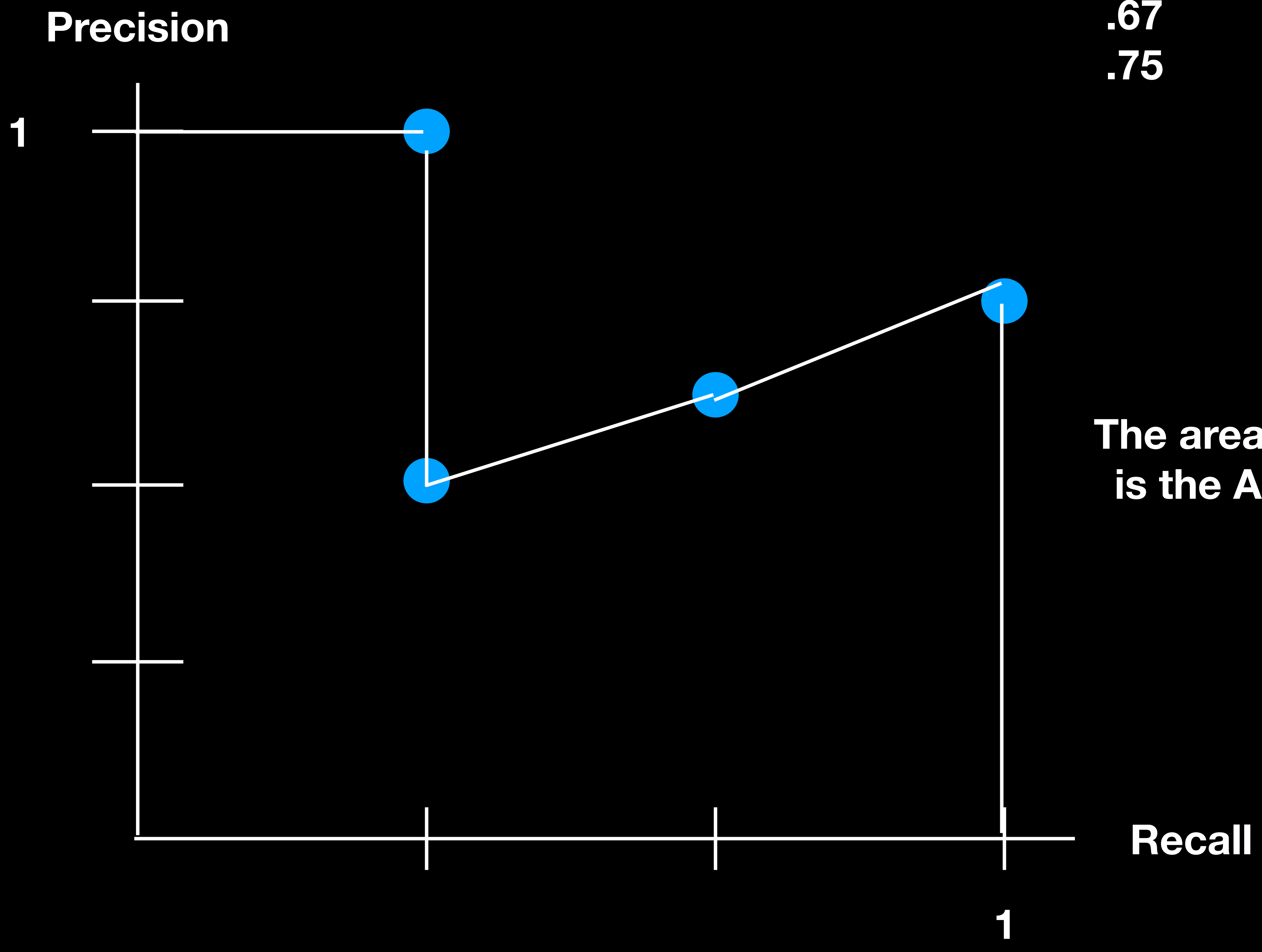
sum of green and blue box areas

green box = ground truth, blue box = estimated truth
(IoU > .5 = detected)

Example

- 3 GT dog boxes, 5 dog boxes from the algorithm following NMS with probabilities .9, .7, .55, .6, .8
- Does .9 probability bounding box match a GT box with IoU > .5?
- Yes: Precision = 1/1, Recall = 1/3. Remove the associated GT box from further consideration.
- Does .8 probability bounding box match any of the remaining GT boxes?
- No: Precision = 1/2, Recall = 1/3
- Does .7 probability bounding box match the remaining GT boxes?
- Yes: Precision = 2/3, Recall = 2/3. Remove the associated GT box from further consideration.
- Does .6 probability bounding box match the remaining GT boxes?
- Yes: Precision = 3/4, Recall = 3/3. Remove the associated GT box from further consideration.

Precision	Recall
1	.33
.5	.33
.67	.67
.75	1



The area under this curve is the AP for the object.

References

- Rapid Object Detection using a Boosted Cascade of Simple Features, Viola and Jones, CVPR 2001
- Histograms of Oriented Gradients for Human Detection, Dalal and Triggs, CVPR 2005
- Cascade Object Detection with Deformable Part Models, Felzenszwalb, et. al., CVPR 2010
- Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, Girshick, et. al., arXiv Nov 2013
- Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, He, et. al., arXiv June 2014
- Fast R-CNN, Girshick, arXiv April 2015
- You Only Look Once: Unified Real Time Object Detection, Redmon, et. al., arXiv June 2015
- Faster R-CNN: Toward Real Time Object Detection with Region Proposals, Rea, et. al., arXiv June 2015
- Single Shot Multi-Box Detector, Liu, et. al., arXiv December 2015
- Mask R-CNN, He, et. al., arXiv March 2017