Anatomy

of

YOLO

v1

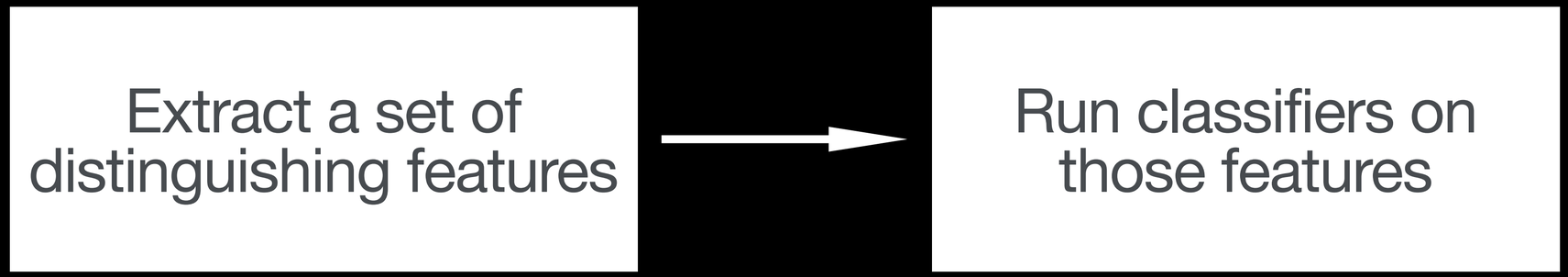
(04/30/2020)

Jihoon Song
jihoonsong.dev@gmail.com

Index

Current Detection Systems

How do they work?



Mainly two steps approach.

Step 1: Extract regions of interest at various locations and scales.

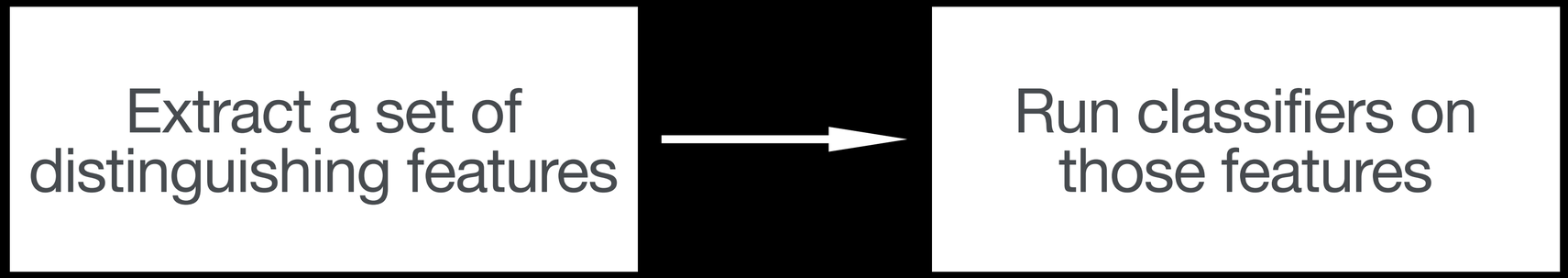
- Sliding window e.g. deformable parts models (DPM)
- Selective search
- Region proposal e.g. R-CNN

Step 2: Run classifiers on them.

Some systems have a post-processing stage to refine their regions of interest.

Current Detection Systems

Drawbacks

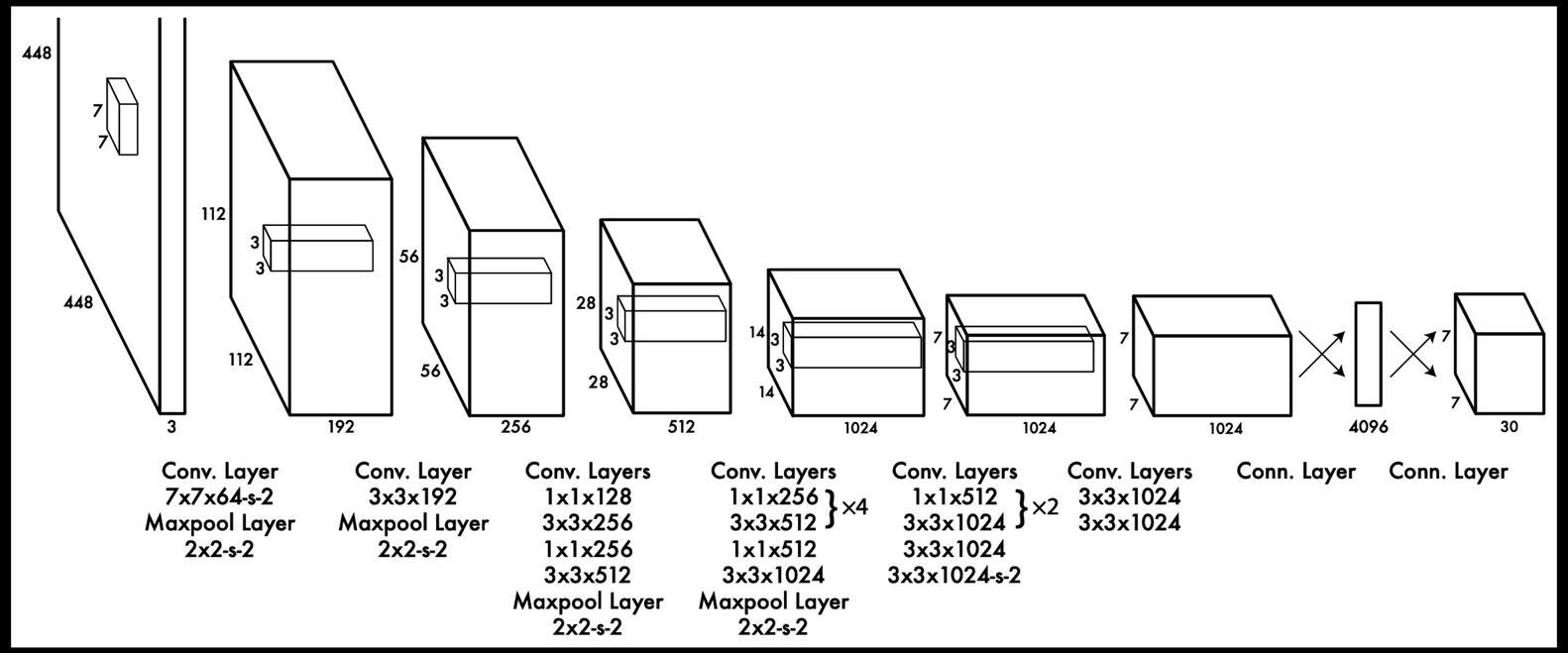


0. Slow
Extracting regions of interest takes a long time.

1. Hard to optimize
Each step is an independent component.
The individual component has to be trained separately.

YOLO

A different approach

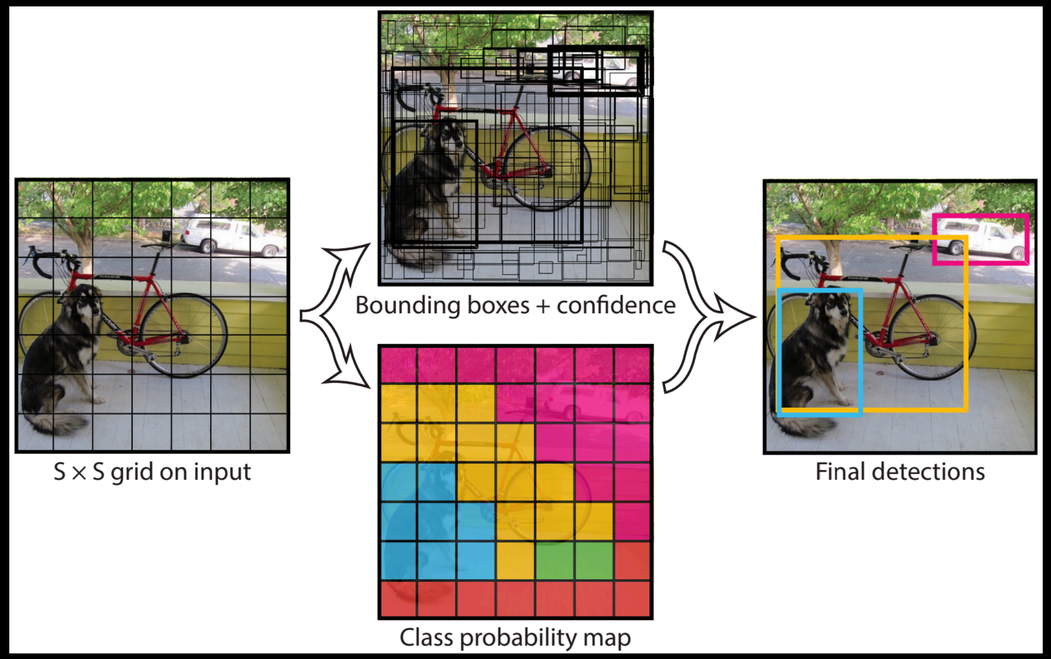


Reframed object detection as a single regression problem.

Use a single network to predict bounding box coordinates and class probabilities.

YOLO

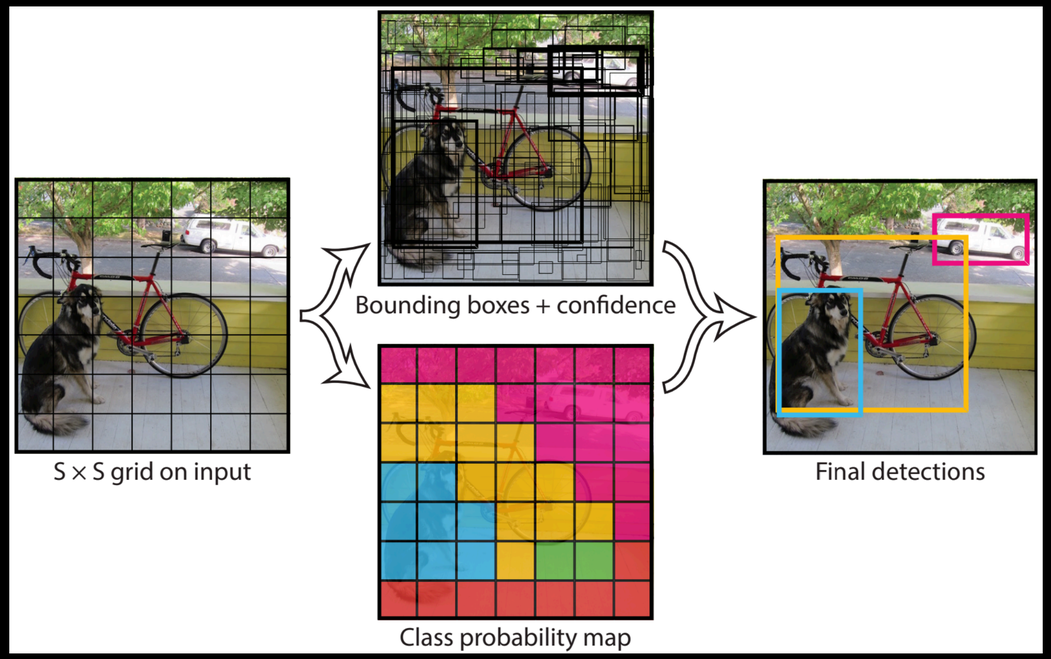
Unified detection



Divides the input image into an S × S grid.
Each grid cell predicts B bounding boxes and confidence scores for those boxes, simultaneously.

YOLO

Unified detection - grid cell



The center of the bounding boxes that a grid cell predicts may or may not reside in the grid cell.
A grid cell is responsible for detecting an object of which the center of the bounding box falls into the grid cell.

YOLO

Unified detection - confidence prediction

Confidence prediction = Pr(Object) * $\text{IOU}_{pred}^{truth}$

$$\text{Pr(Object)} = \begin{cases} 1, & \text{if object exists} \\ 0, & \text{otherwise} \end{cases}$$

How confident the grid cell is that the bounding box contains an object.

YOLO

Unified detection - conditional class probability

Conditional class probability $=$ Pr(Class$_i$ | Object)

How confident the grid cell is that the object is Class$_i$.

YOLO

Unified detection - confidence score

Confidence score = Conditional class probability * Confidence prediction

$$= \Pr(\text{Class}_i \mid \text{Object}) * \Pr(\text{Object}) * \text{IOU}_{pred}^{truth}$$

$$= \Pr(\text{Class}_i) * \text{IOU}_{pred}^{truth}$$

How confident the grid cell is that the bounding box contains an object of Class_i.

YOLO

Unified detection - grid cell

More precisely, each grid cell predicts B bounding boxes and C conditional class probabilities.

Each bounding box consists of 5 predictions;
x, y, w, h, and confidence predictions.
x and y are parametrized to be offsets of a particular grid cell location, i.e. [0, 1].
w and h are normalized by the image width and height, i.e. [0, 1].

Thus, the final prediction is $S \times S \times (B * 5 + C)$ tensor.

A grid cell predicts only one **set** of conditional class probabilities, regardless of the number of bounding boxes it has.

YOLO

Unified detection - grid cell

```
typedef struct detection{
    box bbox;
    int classes;
    float *prob;
    float *mask;
    float objectness;
    int sort_class;
} detection;
```

A data structure of the bounding box.

Each grid cell has an array of these.

YOLO

Unified detection - grid cell

```
typedef struct detection{
    box bbox;
    int classes;
    float *prob;
    float *mask;
    float objectness;
    int sort_class;
} detection;
```

bbox: a bounding box

classes: i of Class$_i$

prob: Pr(Class$_i$ | Object)

mask: the indices of the bounding boxes it is responsible for predicting

objectness: Pr(Object)

sort_class: if j, sort in descending order based on class j

YOLO

Non-maximum suppression - beforehand

Now we have S * S * B bounding boxes total.

Since the center of the bounding boxes that a grid cell predicts may or may not reside in the grid cell, we have to distribute each bounding box to its designated grid cell.

The bounding boxes belong to the same grid cell share one set of confidence class probabilities, which the grid cell has.

YOLO

Non-maximum suppression - within each grid cell

```
void do_nms(box *boxes, float **probs, int total, int classes, float thresh)
{
    int i, j, k;
    for(i = 0; i < total; ++i){
        int any = 0;
        for(k = 0; k < classes; ++k) any = any || (probs[i][k] > 0);
        if(!any) {
            continue;
        }
        for(j = i+1; j < total; ++j){
            if (box_iou(boxes[i], boxes[j]) > thresh){
                for(k = 0; k < classes; ++k){
                    if (probs[i][k] < probs[j][k]) probs[i][k] = 0;
                    else probs[j][k] = 0;
                }
            }
        }
    }
}
```

Non-maximum suppression among bounding boxes within each grid cell.

After this non-maximum suppression, each grid cell has one set of confidence class probabilities, each of which is the highest probability among those bounding boxes.

YOLO

Non-maximum suppression - within the input image

```
void do_nms_obj(detection *dets, int total, int classes, float thresh)
{
    int i, j, k;
    k = total - 1;
    for (i = 0; i <= k; ++i) {
        if (dets[i].objectness == 0) {
            detection swap = dets[i];
            dets[i] = dets[k];
            dets[k] = swap;
            --k;
            --i;
        }
    }
    total = k + 1;

    for (i = 0; i < total; ++i) {
        dets[i].sort_class = -1;
    }

    qsort(dets, total, sizeof(detection), nms_comparator_v3);
    for (i = 0; i < total; ++i) {
        if (dets[i].objectness == 0) continue;
        box a = dets[i].bbox;
        for (j = i + 1; j < total; ++j) {
            if (dets[j].objectness == 0) continue;
            box b = dets[j].bbox;
            if (box_iou(a, b) > thresh) {
                dets[j].objectness = 0;
                for (k = 0; k < classes; ++k) {
                    dets[j].prob[k] = 0;
                }
            }
        }
    }
}
```

Non-maximum suppression among grid cells within the input image.

YOLO

Unified detection - hyperparameters

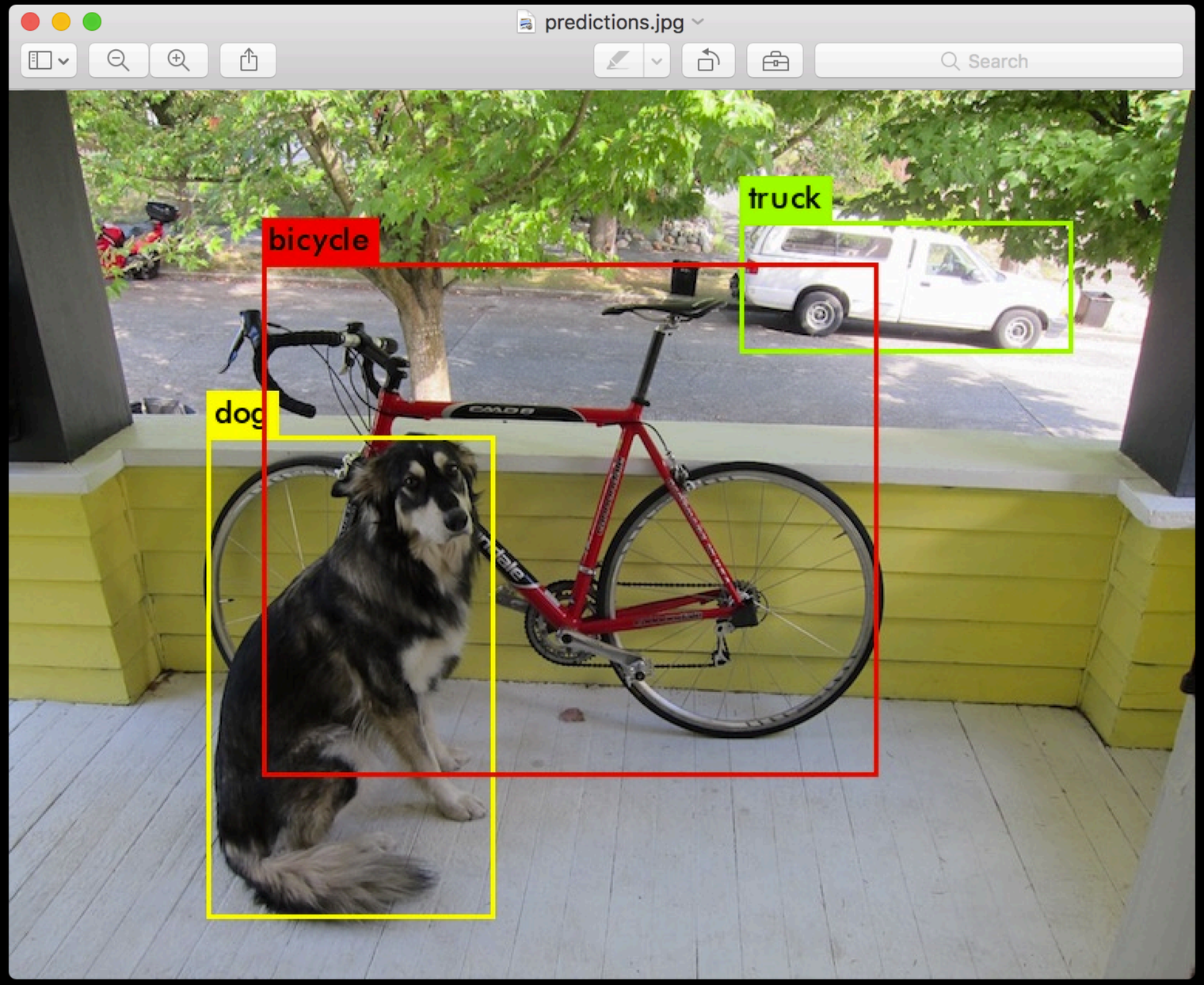
For PASCAL VOC, $S = 7$ and $B = 2$ are used.

PASCAL VOC has 20 labelled classes so $C = 20$.

Thus, the final prediction is a $7 \times 7 \times 30$ tensor.

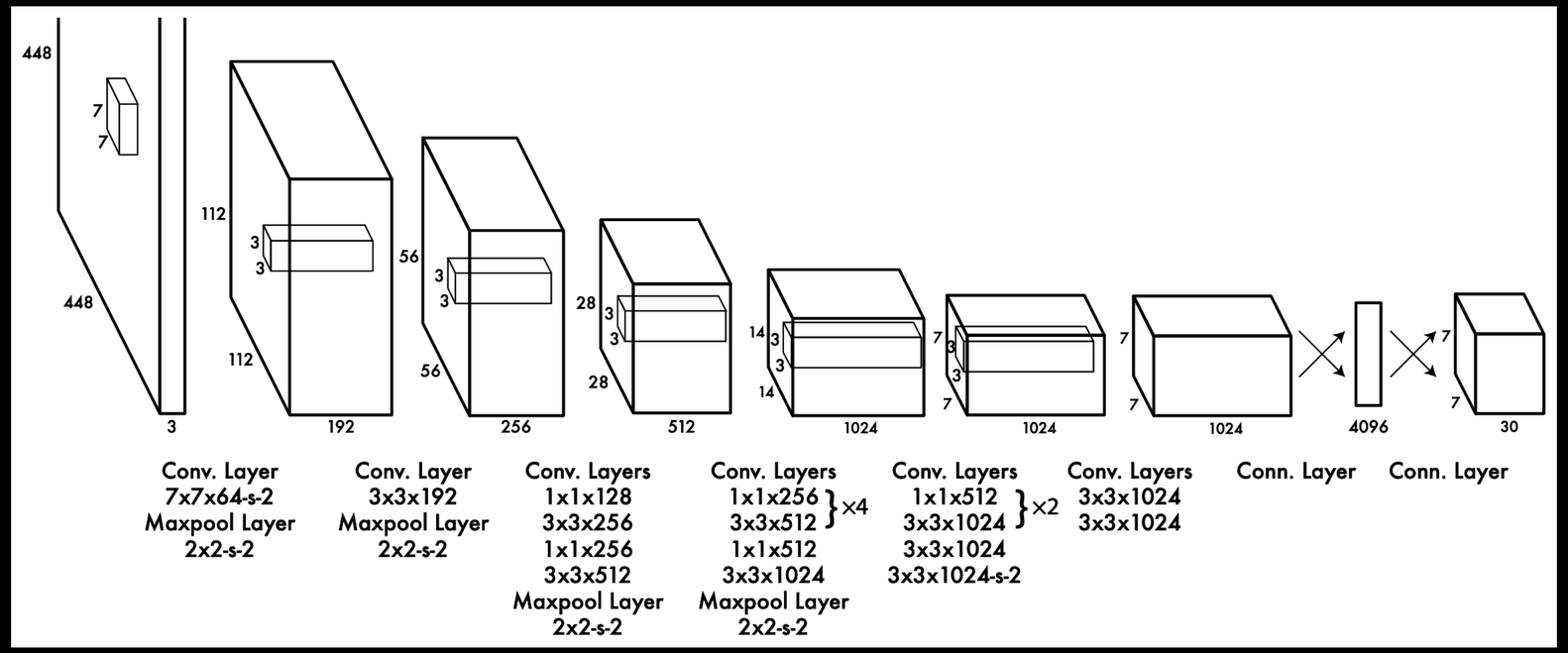
YOLO

Result



YOLO

Network Design

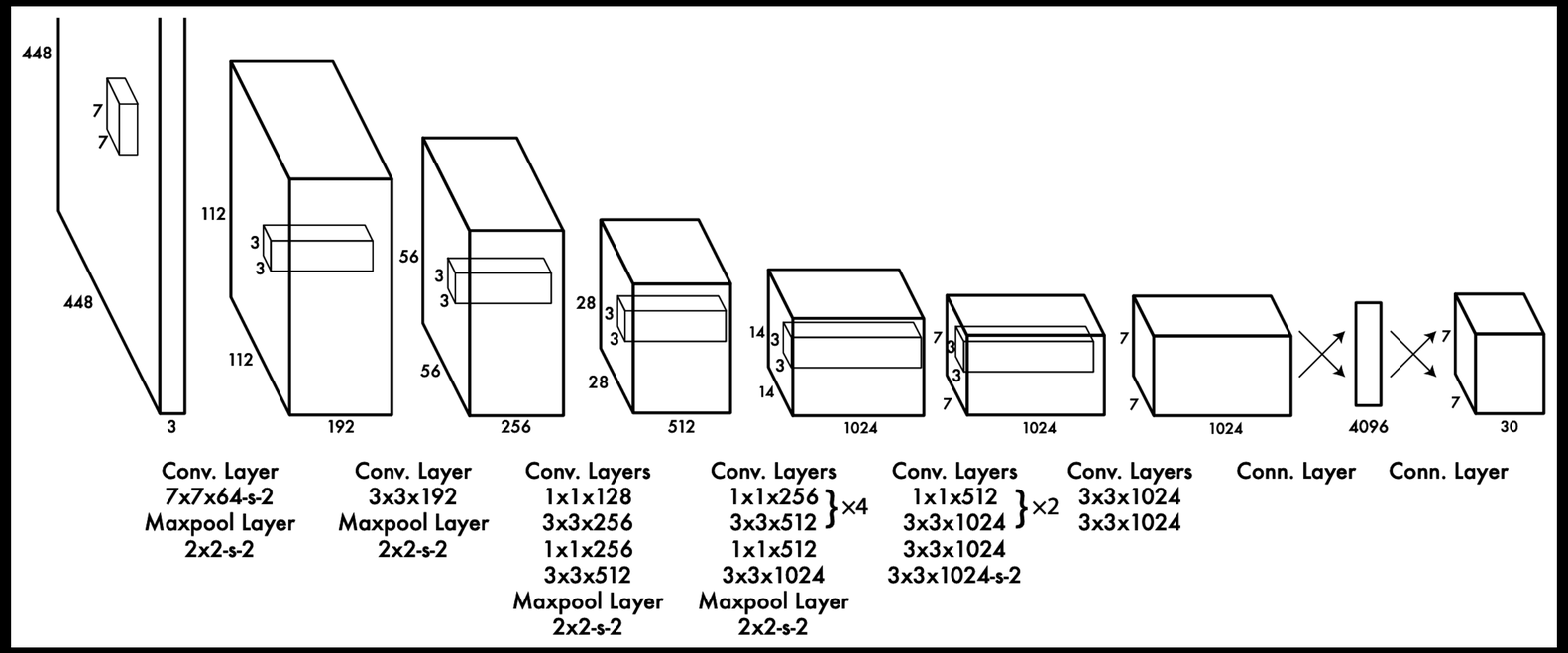


The initial convolutional layers of the network extract features from an image while the fully connected layers predict the output coordinates and probabilities.

In this way, object detection becomes a single regression problem.

YOLO

Network Design - YOLO



YOLO architecture is inspired by the GoogLeNet.

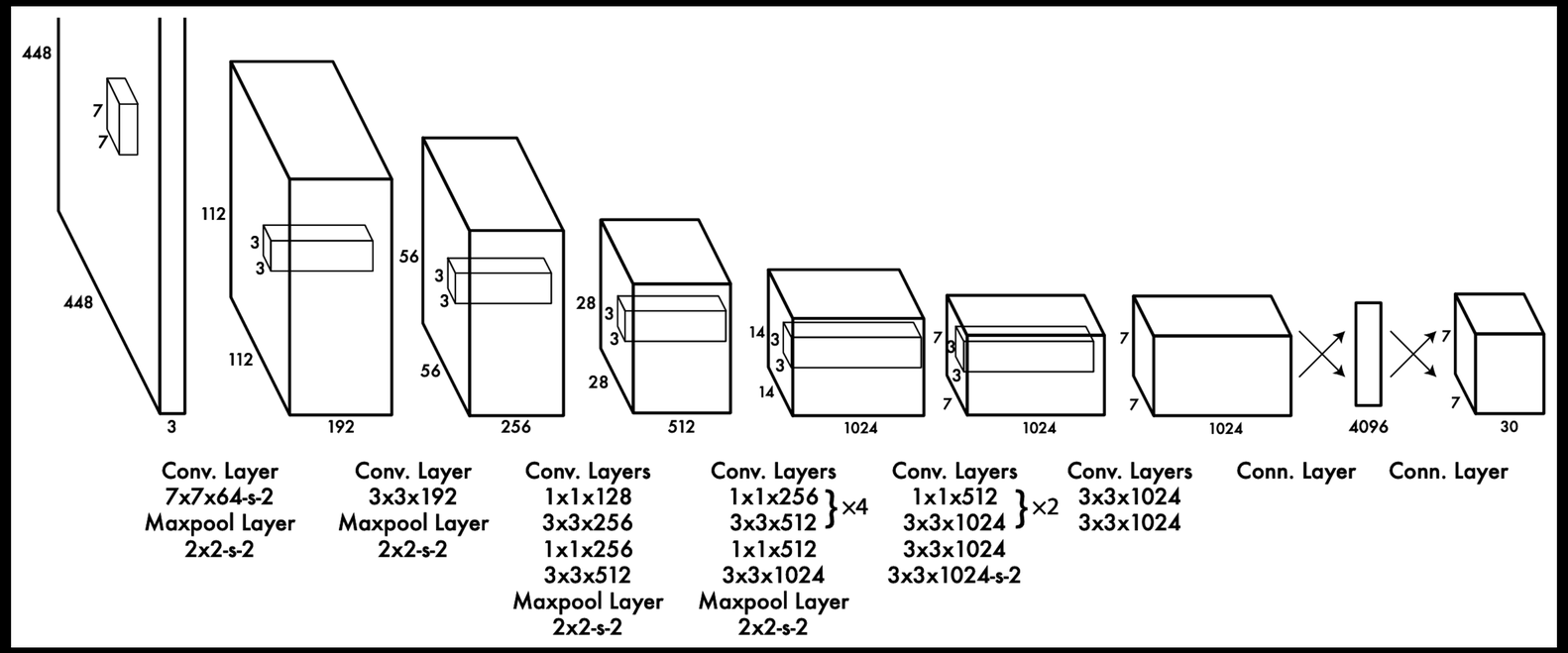
24 convolutional layers followed by 2 fully connected layers.

Instead of the inception modules used by GoogLeNet, YOLO uses 1 × 1 reduction layers followed by 3 × 3 convolutional layers.

YOLO v1 configuration

YOLO

Network Design - Fast YOLO



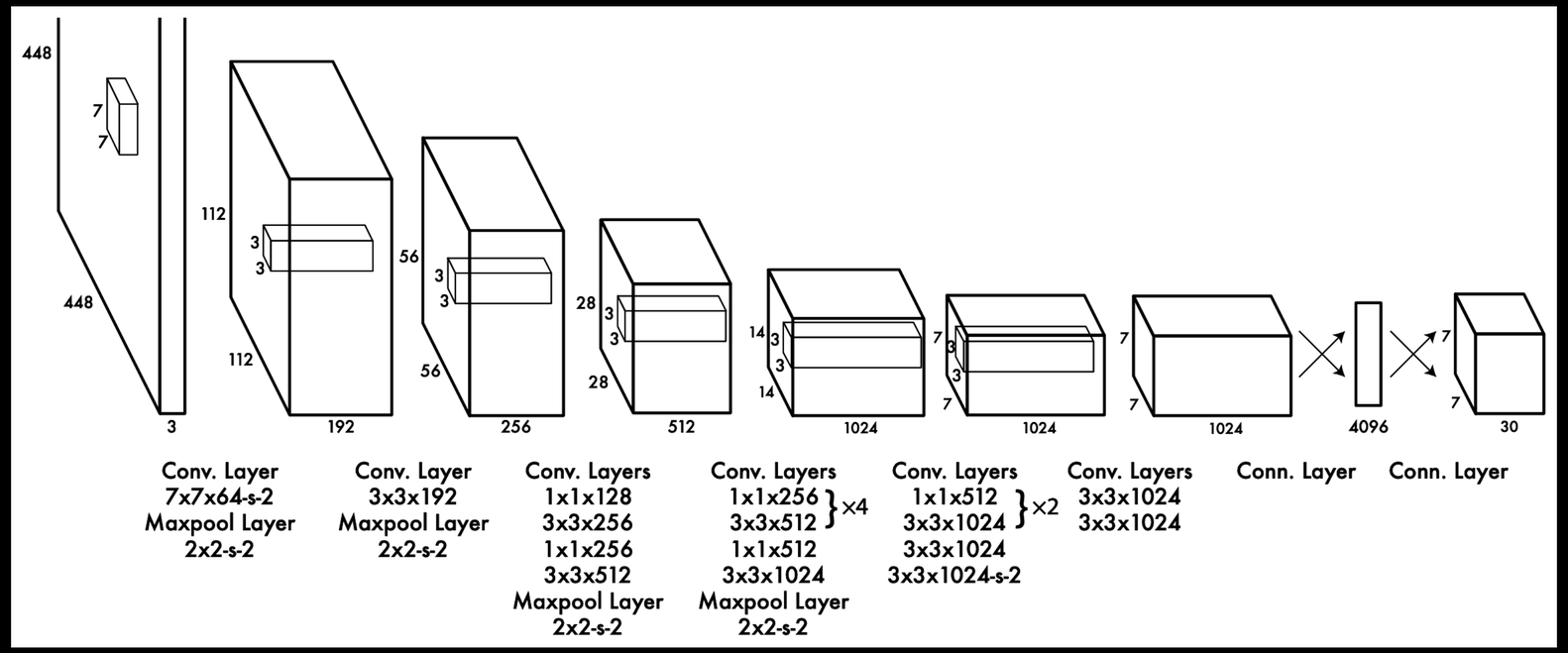
Fast YOLO has fewer convolutional layers (9 instead of 24) and fewer filters in those layers.

Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.

Fast YOLO v1 configuration

YOLO

Training

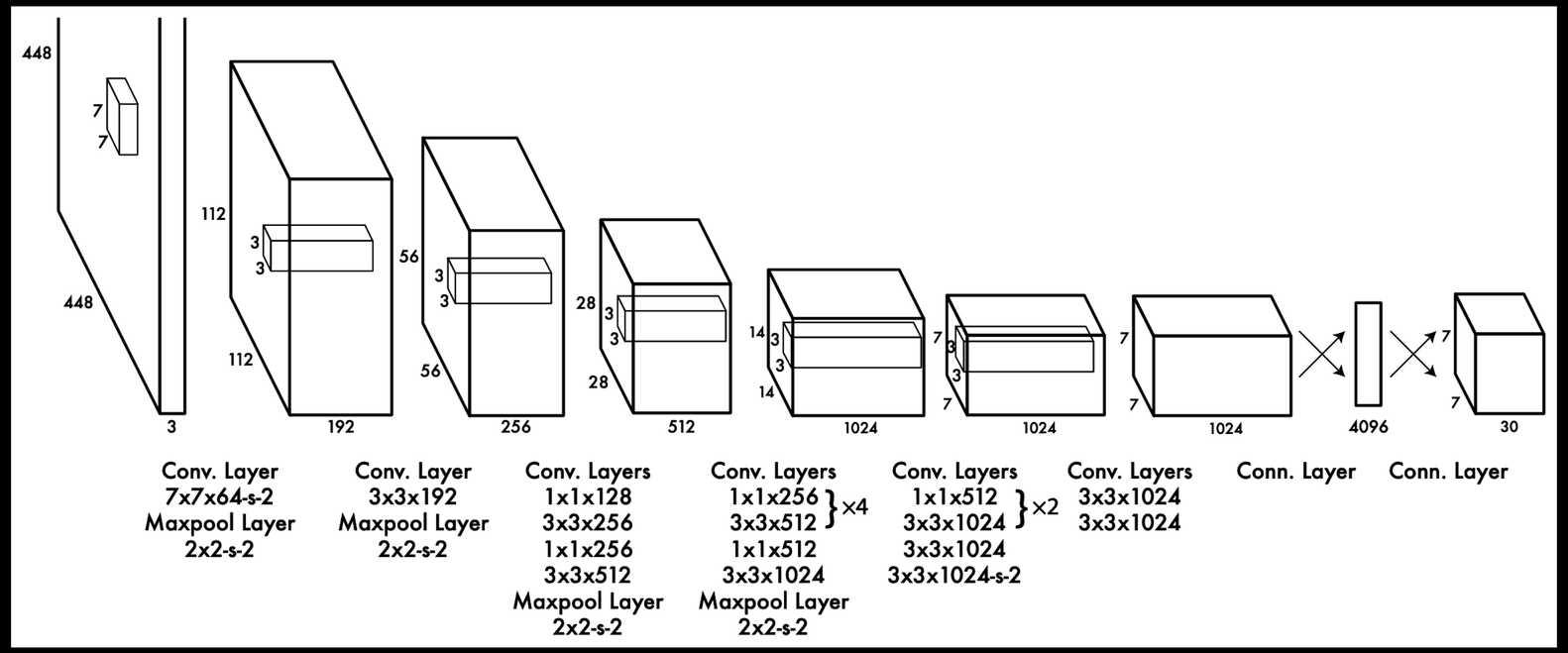


Used the Darknet framework for all training and inference.

The Darknet is an open source neural network framework written in C and CUDA.

YOLO

Training - pretraining

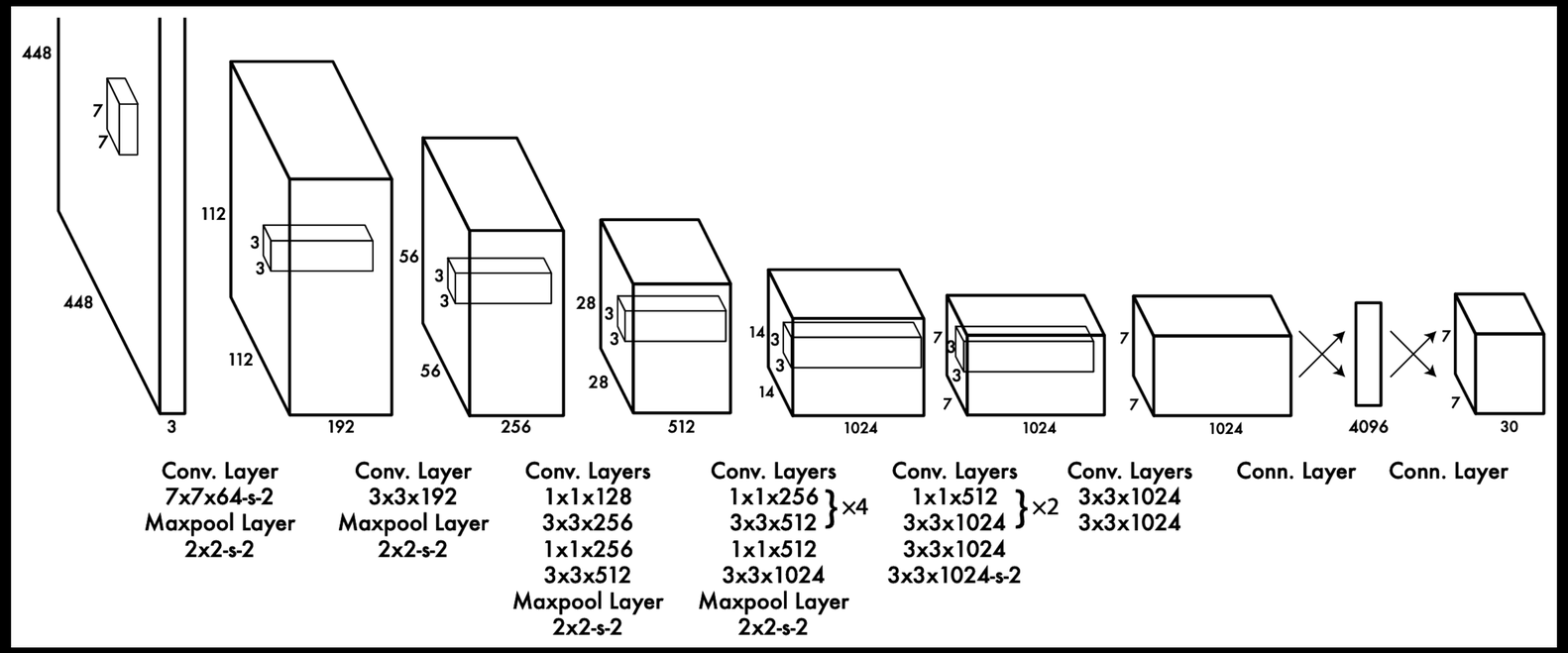


Pretrained the first 20 convolutional layers followed by an average-pooling layer and a fully connected layer on the ImageNet 1000-class competition dataset. Achieved a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set after training for approximately a week.

The input resolution is 224×224.

YOLO

Training - at training



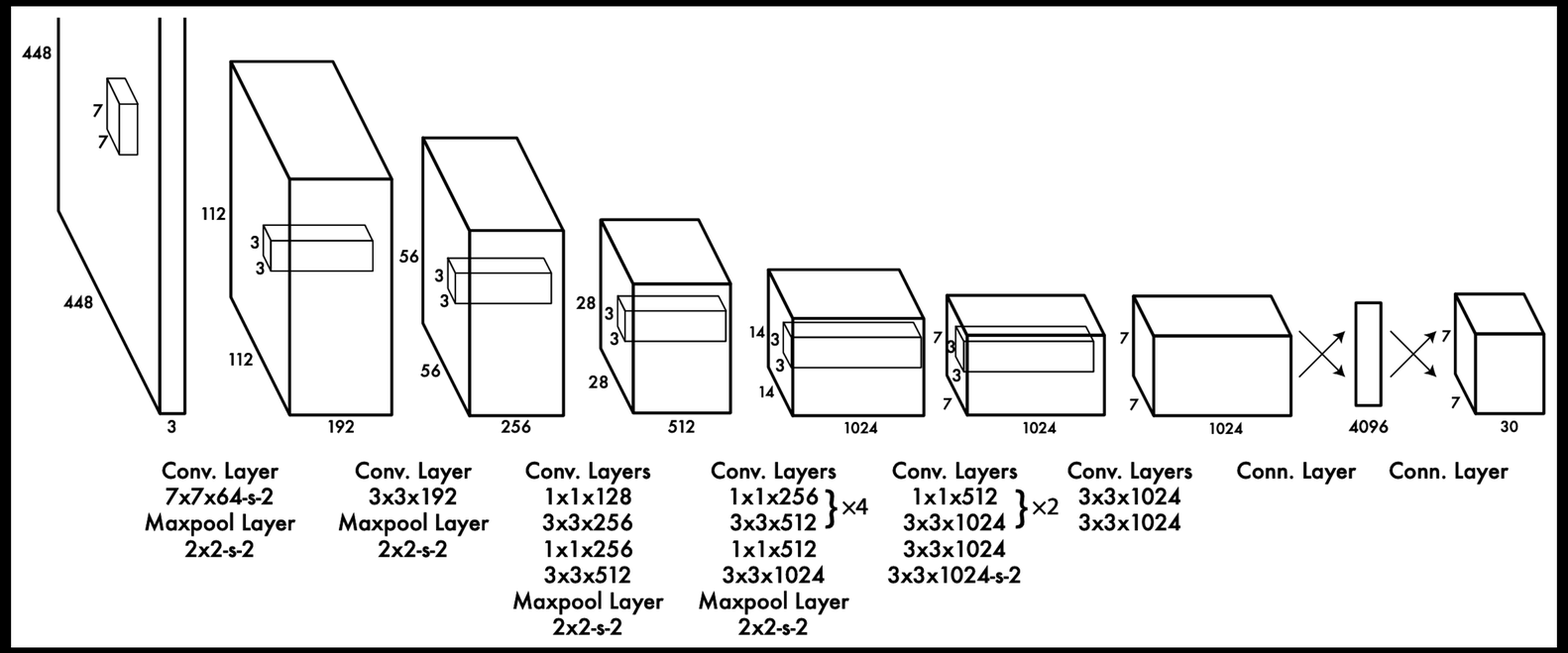
To convert the model to perform detection, added four convolutional layers and two fully connected layers with randomly initialized weights.

Ren et al. shows that adding both convolutional and fully connected layers to pertained networks can improve performance.

The input resolution is 448×448.

YOLO

Training - an activation function



All layers use the following leaky rectified linear activation function.

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$

YOLO

Training - a loss function

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
$$

The loss function has multiple parts.

YOLO

Training - a loss function

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

$$\mathbf{1}_{ij}^{obj} = \begin{cases} 1, & \text{if object exists in } j\text{th bounding box predictor in cell } i \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{1}_{ij}^{noobj} = \begin{cases} 1, & \text{if object not exist in } j\text{th bounding box predictor in cell } i \\ 0, & \text{otherwise} \end{cases}$$

Used sum-squared error because it is easy to optimize.
However, it weights localization error equally with classification error.

YOLO

Training - a loss function

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

$$\mathbf{1}_{ij}^{obj} = \begin{cases} 1, & \text{if object exists in } j\text{th bounding box predictor in cell } i \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{1}_{ij}^{noobj} = \begin{cases} 1, & \text{if object not exist in } j\text{th bounding box predictor in cell } i \\ 0, & \text{otherwise} \end{cases}$$

Also, in every image many grid cells do not contain any object.
This pushes the confidence score of the grid cell that do not contain any object towards zero, often overpowering the gradient from the grid cells that do contain objects.
This can lead to model instability, causing training to diverge early on.

YOLO

Training - a loss function

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

$$\mathbf{1}_{ij}^{obj} = \begin{cases} 1, & \text{if object exists in } j\text{th bounding box predictor in cell } i \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{1}_{ij}^{noobj} = \begin{cases} 1, & \text{if object not exist in } j\text{th bounding box predictor in cell } i \\ 0, & \text{otherwise} \end{cases}$$

To remedy this, two parameters λ_{coord} and λ_{noobj} were introduced to increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects.

The authors put $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = 0.5$.

YOLO

Training - a loss function

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

$$\mathbf{1}_{ij}^{obj} = \begin{cases} 1, & \text{if object exists in } j\text{th bounding box predictor in cell } i \\ 0, & \text{otherwise} \end{cases}$$

Sum-squared error also equally weights errors in large boxes and small boxes. However, this error metric should reflect that small deviations in large boxes matter less than in small boxes.

To remedy this, each grid cell predicts the square root of the bounding box width and height instead of the width and height directly.

YOLO

Training - a loss function

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

$$\mathbf{1}_{ij}^{obj}=\begin{cases}1, & \text{if object exists in } j\text{th bounding box predictor in cell } i\\ 0, & \text{otherwise}\end{cases}$$

$$\mathbf{1}_{i}^{obj}=\begin{cases}1, & \text{if object exists in cell } i\\ 0, & \text{otherwise}\end{cases}$$

After the non-maximum suppression, only one bounding box predictor is responsible for each object.

To reflect this, assign one predictor to be responsible for predicting an object based on which prediction has the highest current IOU with the ground truth.

YOLO

Training - hyperparameters

135 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012.
Batch size of 64, a momentum of 0.9 and a decay of 0.0005.

The model often diverges if it starts at a high learning rate due to unstable gradients.
For the first epochs raise the learning rate from 10^{-3} to 10^{-2}.
Continue training with 10^{-2} for 75 epochs, then 10^{-3} for 30 epochs, and finally 10^{-4} for 30 epochs.

A dropout layer with rate = 0.5 after the first fully connected layer prevents co-adaptation between the fully connected layers.

Random scaling and translations of up to 20% of the original image size.
Randomly adjust the exposure and saturation to the image by up to a factor of 1.5 in the HSV color space.

Experiments

Comparison to other real-time systems

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

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

Fast YOLO is the fastest real-time object detector on PASCAL with 52.7% mAP.
YOLO is the most accurate real-time object detector with 63.4% mAP.
Only Sadeghi et al. produces a real-time object detector but with lower mAP than YOLO.

Experiments

VOC 2007 error analysis

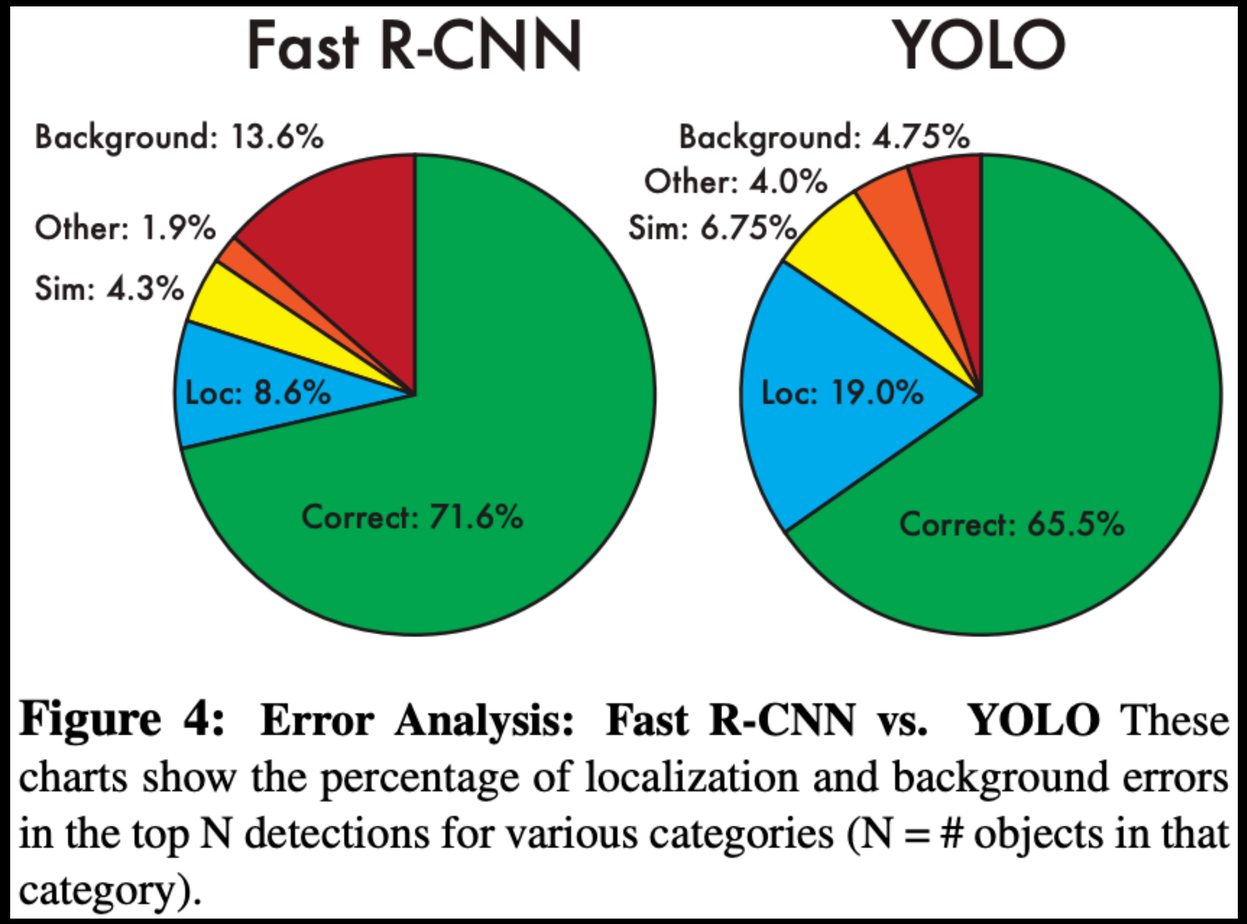


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

Comparison to Fast R-CNN, one of the highest performing detectors on PASCAL.

Experiments

VOC 2007 error analysis

Analyze the top N predictions for each category at test time.
Each prediction is either correct or classified based on the type of error.

Correct: correct class and IOU > 0.5

Localization: correct class, 0.1 < IOU < 0.5

Similar: class is similar, IOU > 0.1

Other: class is wrong, IOU > 1

Background: IOU < 0.1 for any object

YOLO struggles to localize objects correctly.
Localization errors account for more of YOLO's errors than all others.
Fast R-CNN makes much fewer localization errors but far more background error.

Experiments

Combining Fast R-CNN and YOLO

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

Fast R-CNN has a higher AP on PASCAL than YOLO.
YOLO makes fewer background errors compared to Fast R-CNN.
Combining these two models would give us a significant boost in performance.

Experiments

Combining Fast R-CNN and YOLO

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

If two models predict similar boxes, make a boost based on their probabilities and IOU.

Unfortunately, the ensemble doesn't benefit from the speed of YOLO since each model runs separately.

Experiments

VOC 2012 results

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

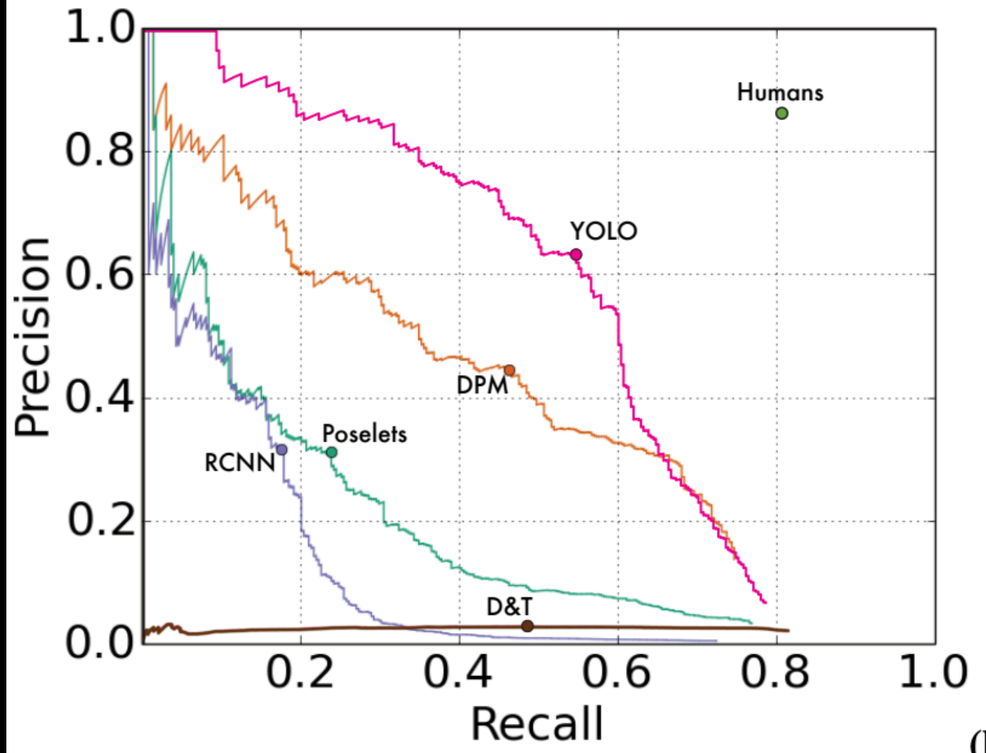
Table 3: PASCAL VOC 2012 Leaderboard. YOLO compared with the full `comp4` (outside data allowed) public leaderboard as of November 6th, 2015. Mean average precision and per-class average precision are shown for a variety of detection methods. YOLO is the only real-time detector. Fast R-CNN + YOLO is the forth highest scoring method, with a 2.3% boost over Fast R-CNN.

YOLO struggles with small objects such as bottle, sheep, and tv.

Combined Fast R-CNN + YOLO model is one of the highest performing detector.

Experiments

Generalizability: person detection in artwork



(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.

Figure 5: Generalization results on Picasso and People-Art datasets.

Comparison to other detectors on the Picasso Dataset and the People-Art Dataset, which are two datasets for testing person detection on artwork.

Experiments

Generalizability: person detection in artwork

R-CNN has high AP on VOC 2007 but its AP drops considerably when applied to the artwork.
R-CNN only sees small regions in the classifier step and thus requires good proposals.
However, R-CNN uses the selective search for bounding box proposals, which is tuned for natural images, not artwork.

DPM maintains its AP well when applied to the artwork.
DPM has strong spatial models of the shape and layout of objects.
However, DPM has lower AP on VOC 2007 than R-CNN and YOLO.

YOLO has the highest AP on VOC 2007 and its AP degrades less when applied to the artwork than other detectors.
YOLO models the size and shape of objects, as well as relationships between objects and where objects commonly appear.

YOLO

Pros

0. Extremely fast
    - Process images in real-time at 45 frames per second.
1. End-to-end training
    - It is a single network.
2. Fewer mistakes of background false positives
    - Performs detection based on the feature maps from the entire image.
    - Makes less than half # background errors compared to Fast R-CNN.
2. Generalizes to new domains better than other detectors
    - Outperforms other detectors when applied to other domains like artwork.

YOLO

Cons

0. Has strong spatial constraints
    - Each grid cell only predicts two boxes and can only have one class.
    - Struggles to localize small objects correctly, especially with small ones.
    - Struggles to localize small objects that appear in groups.
1. Susceptible to localization errors
    - Less accurate than state-of-the-art detectors.

Remark

The grid approach to bounding box prediction is based on the MultiGrasp system for regression to grasps by Redmon et al.

YOLO v2 improves accuracy while making it faster.
YOLO v3 has incremental improvement at the cost of speed.
YOLO v4 is released just a week before. (on 04/23/2020)

Materials

YOLO v1 paper and code
YOLO v2 paper and code
YOLO v3 paper and code
YOLO v4 paper and code

Thank you!