Object Detection

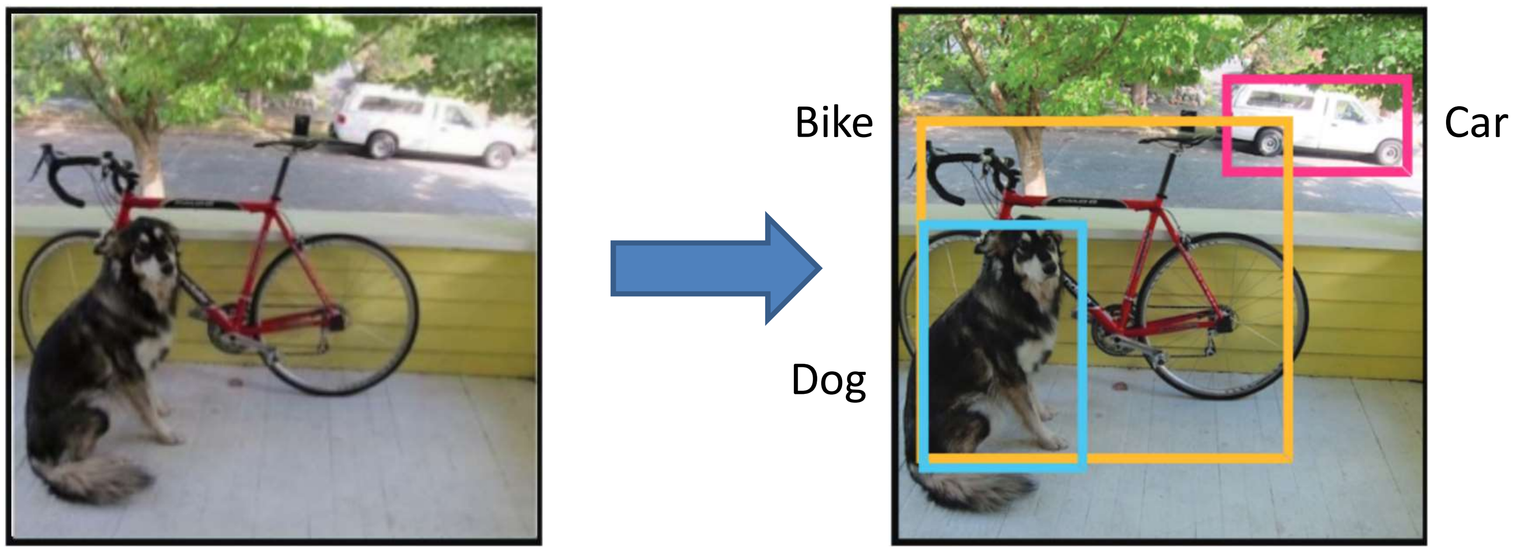
Advanced Computer Vision

Niels Landwehr

Overview

Problem Setting Detection

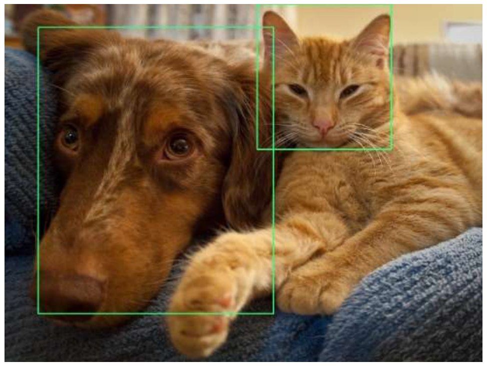
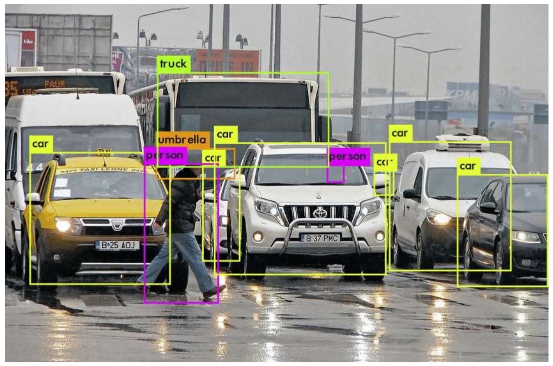
- Problem setting object detection: given image, find and classify objects
  - Input: image
  - Output: set of bounding boxes with class labels (from a fixed set of classes the model has been trained to recognize)

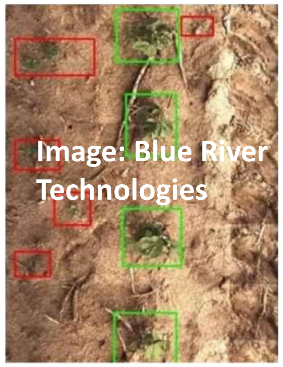


Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Training Data For Object Detection

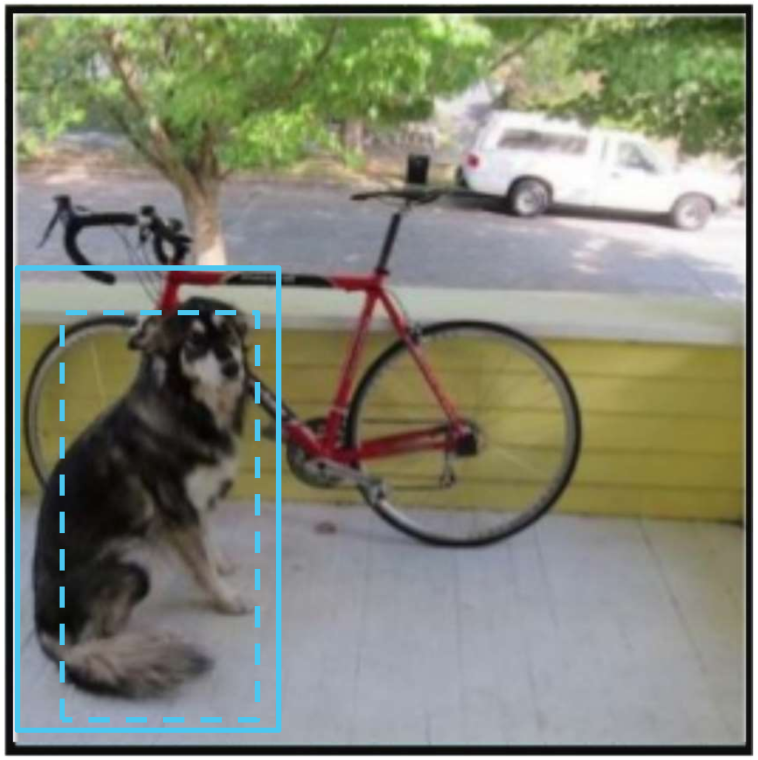
- Training data for object detection: images that have been manually annotated with ground truth bounding boxes
- Many application domains from autonomous driving to robotics, agriculture, tagging objects in photo collections, ...
- Often used to track objects in videos (e.g. traffic)



Performance Measure IOU

- How correct is a bounding box? Exactly matching ground-truth box is unrealistic.
- Intersection over union (IOU) measure: gradual measure of how well a predicted bounding box agrees with a ground truth bounding box



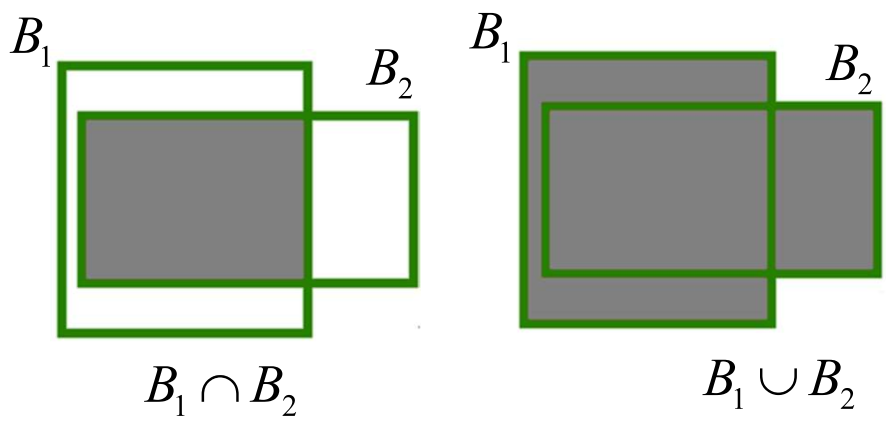
How well does the prediction (dashed) match ground truth (solid)?

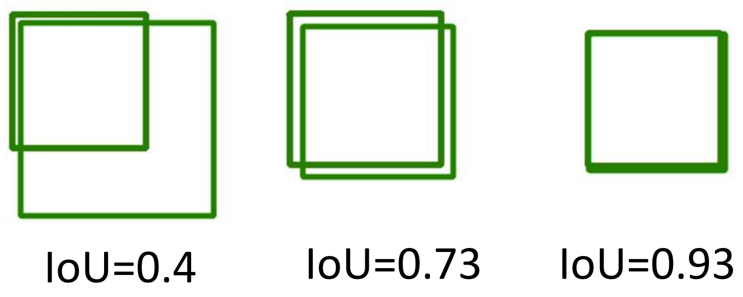
Performance Measure IOU

- How correct is a bounding box? Exactly matching ground-truth box is unrealistic.
- Intersection over union (IoU) measure: denote the ground truth bounding box by B_1, the predicted bounding box by B_2 . Define

$$IoU(B_1, B_2) = \frac{B_1 \cap B_2}{B_1 \cup B_2}$$

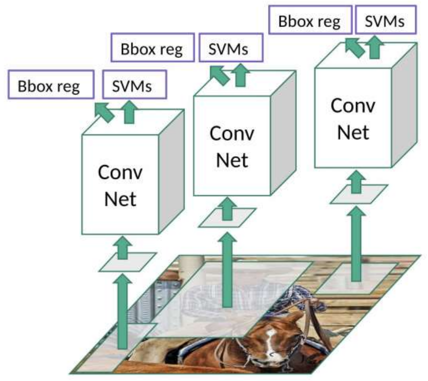
- IoU is symmetric, IoU=0 if no overlap, IoU=1 if boxes are identical





Overview Object Detection Models

- Deep neural networks for object detection build upon convolutional neural network architectures, but use specific architecture features and loss functions to match the particular problem setting
- A relatively large number of models have been developed:
  - „R-CNN“-style models: R-CNN, Fast R-CNN, Faster R-CNN



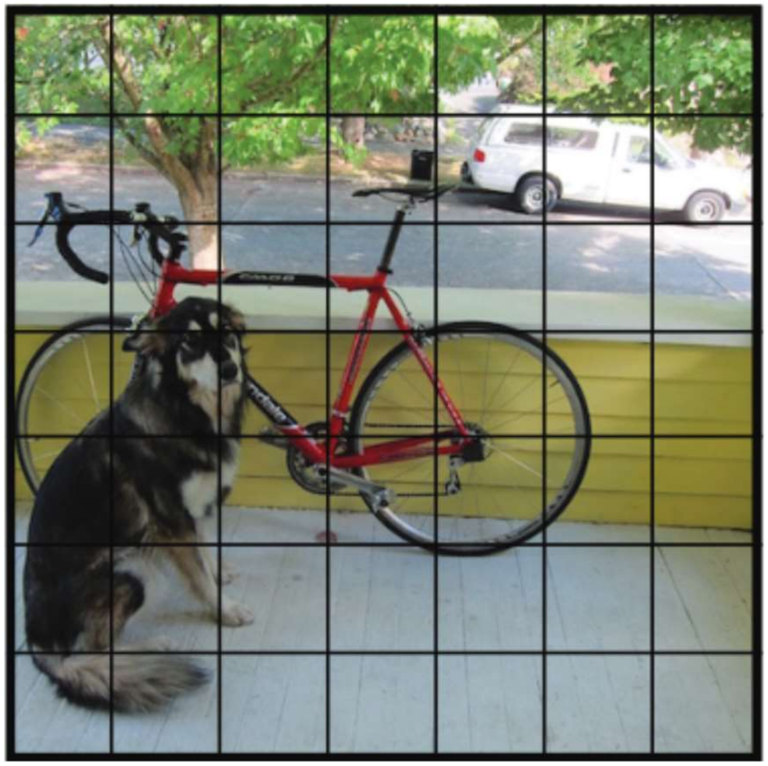
„Regions with CNN features“
State-of-the-art accuracy, but relatively complex and slow

  - SSD („single shot detector“): faster than R-CNNs, slightly less accurate
  - **YOLO („you only look once“): simpler, even faster than SSD, comparable accuracy.** We specifically discuss YOLOv1 [Redmon et al., 2016].

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Idea of YOLO Model: Cell-based Predictions

- **How can a neural network model predict a set of bounding boxes?**
- Idea of YOLO model: conceptually split the image into grid cells. For each grid cell, predict bounding boxes of possible objects centered at that grid cell



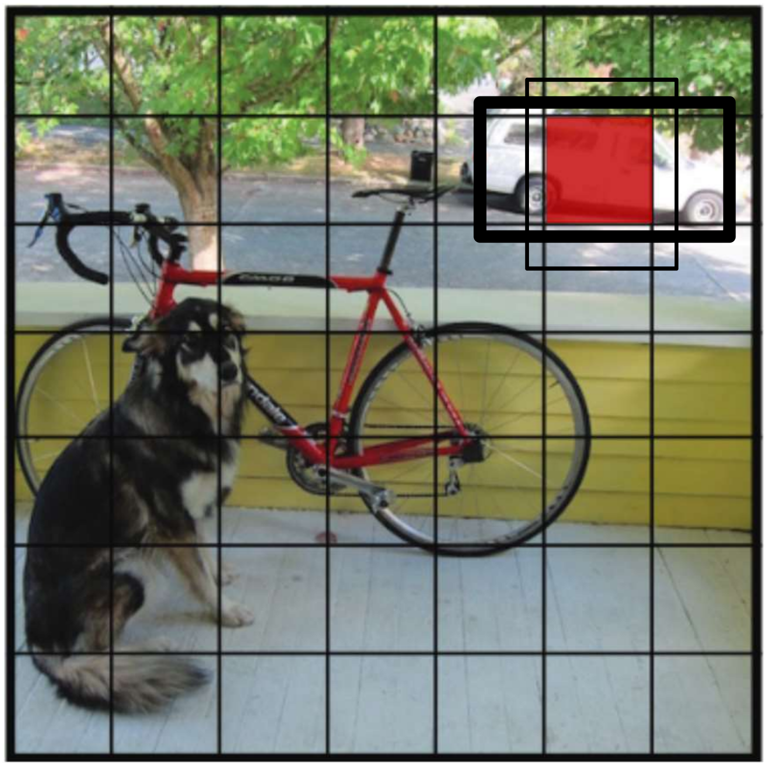
Split image into *S* x *S* grid cells.
S = 7 most frequent choice.

Note: Image is not actually split
into single patches, this is just conceptual.

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Predict Bounding Boxes at Grid Cells

- For each cell, predict *B* bounding boxes with confidence scores. Boxes are encoded by their center coordinates *x*, *y* and their width and height *w*, *h*



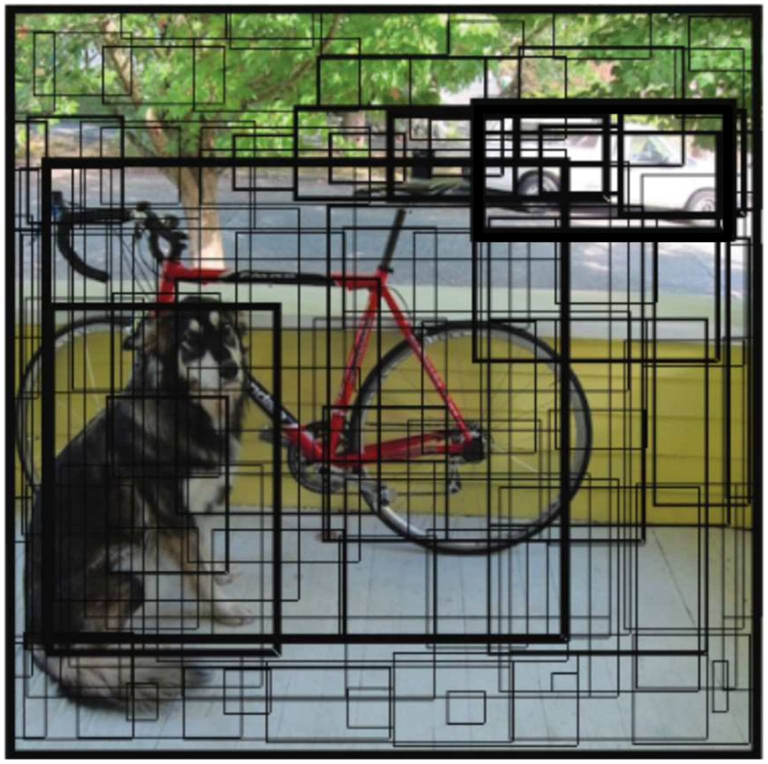
Red cell predicts *B*=2 bounding boxes that have their center in cell

Each bounding box is encoded by 5 numbers:

Box center *x* (relative to cell boundaries)
Box center *y* (relative to cell boundaries)
Box width *w* (relative to image size)
Box height *h* (relative to image size)
Confidence score *C*: how likely that the bounding box actually contains an object?

Bounding Boxes Predicted in Complete Image

- As there are a fixed number of cells and a fixed number of predicted bounding boxes per cell, the model always predicts fixed number of boxes
- Most of these do **not** contain objects, and should have low confidence scores



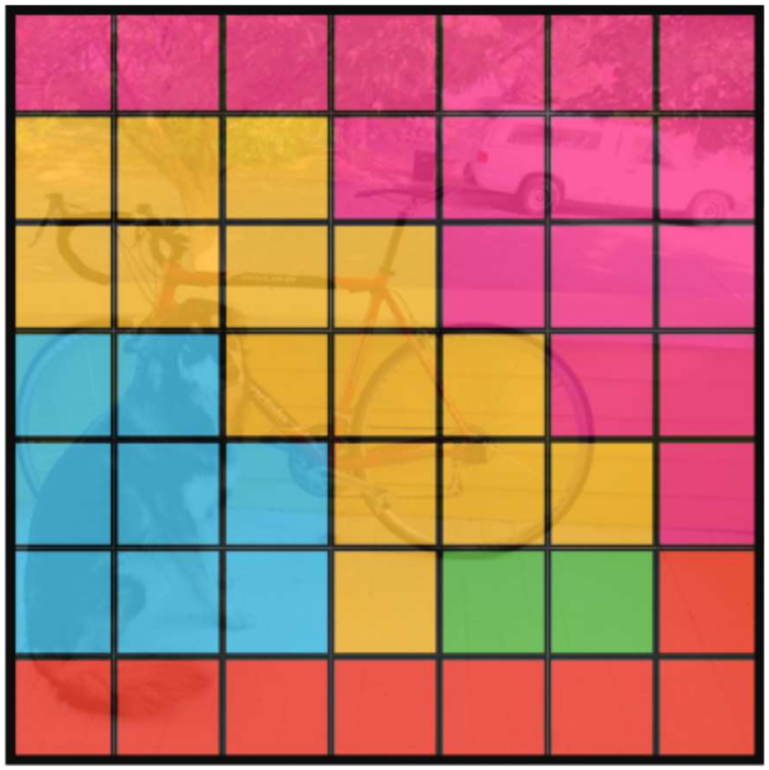
Only a few of the predicted bounding boxes contain actual objects.

These are visible by their high confidence scores.

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Predicted Class Probabilities

- Additionally, model predicts distribution $p(c)$ over classes for each cell
- Interpretation: if there is an object centered at this cell, what is the probability that it belongs to class *c*?



Probability map corresponds to coarse segmentation of image

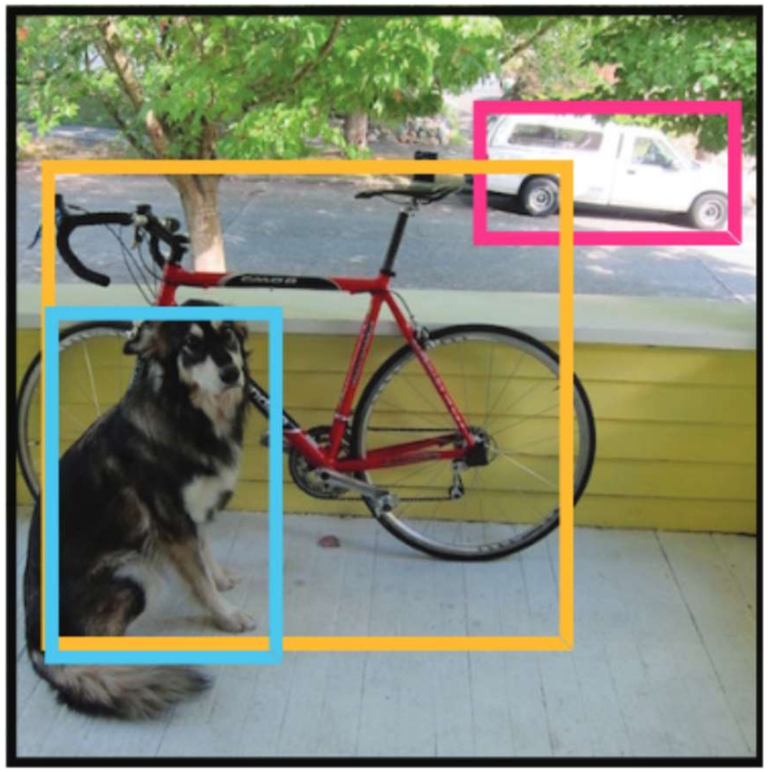
Probabilities are conditioned on object being present:

$$p(c) = p(\text{class} = c \mid \text{there is an object centered at cell})$$

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Final Predictions

- Obtaining final object detections:
  - Assign bounding boxes to classes according to class map
  - Only keep bounding boxes with large confidence, non-max-suppression



Only keep those bounding boxes that exceed a fixed threshold (hyperparameter)

Label each bounding box according to class map

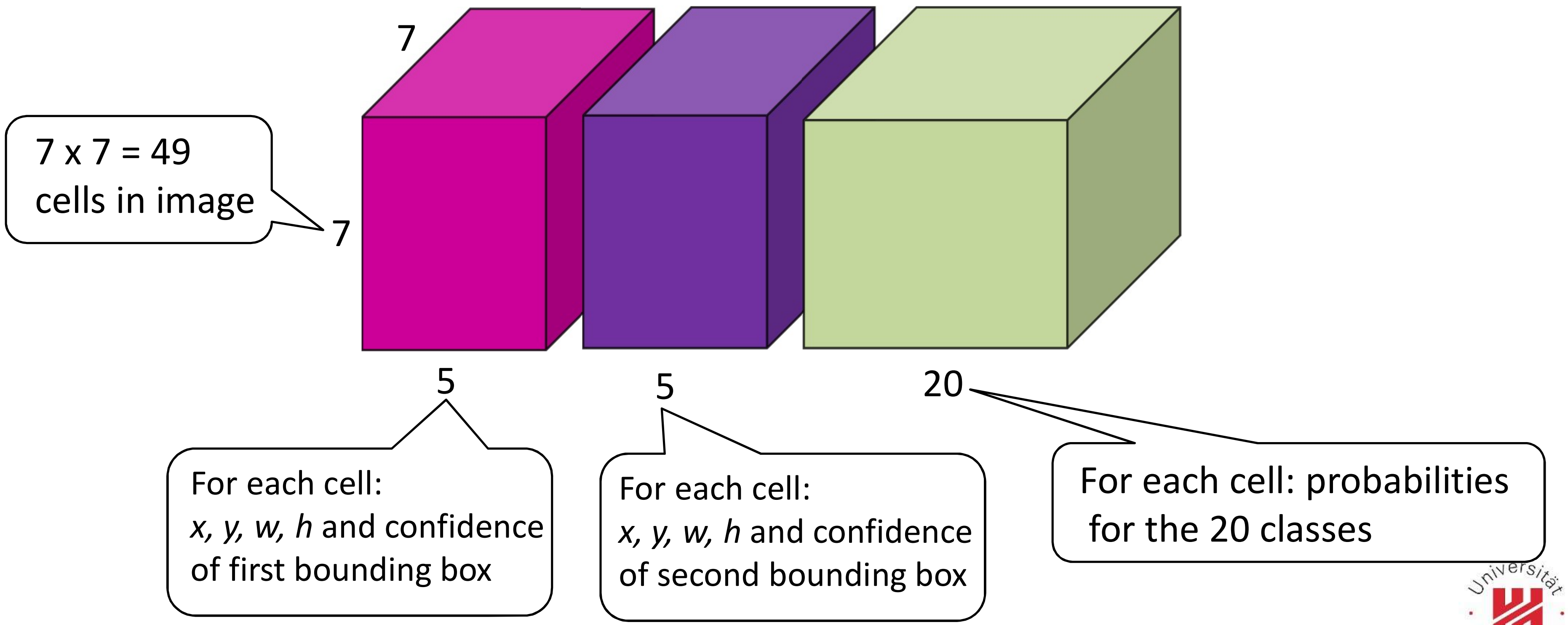
Non-max-suppression: if similar bounding boxes are predicted next to each other, keep the one with higher confidence

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Formal Model Output

- More formally: input and output of YOLO model?
- **Input**: image of fixed size (3D tensor)
- **Output**: 3D-Tensor of size $S \times S \times (B \cdot 5 + k)$, where k is the number of classes

  Standard: $S = 7, B = 2, k = 20,$ overall output size 1470 values


7 x 7 = 49
cells in image
7
7
5
5
20
For each cell: x, y, w, h and confidence of first bounding box
For each cell: x, y, w, h and confidence of second bounding box
For each cell: probabilities for the 20 classes

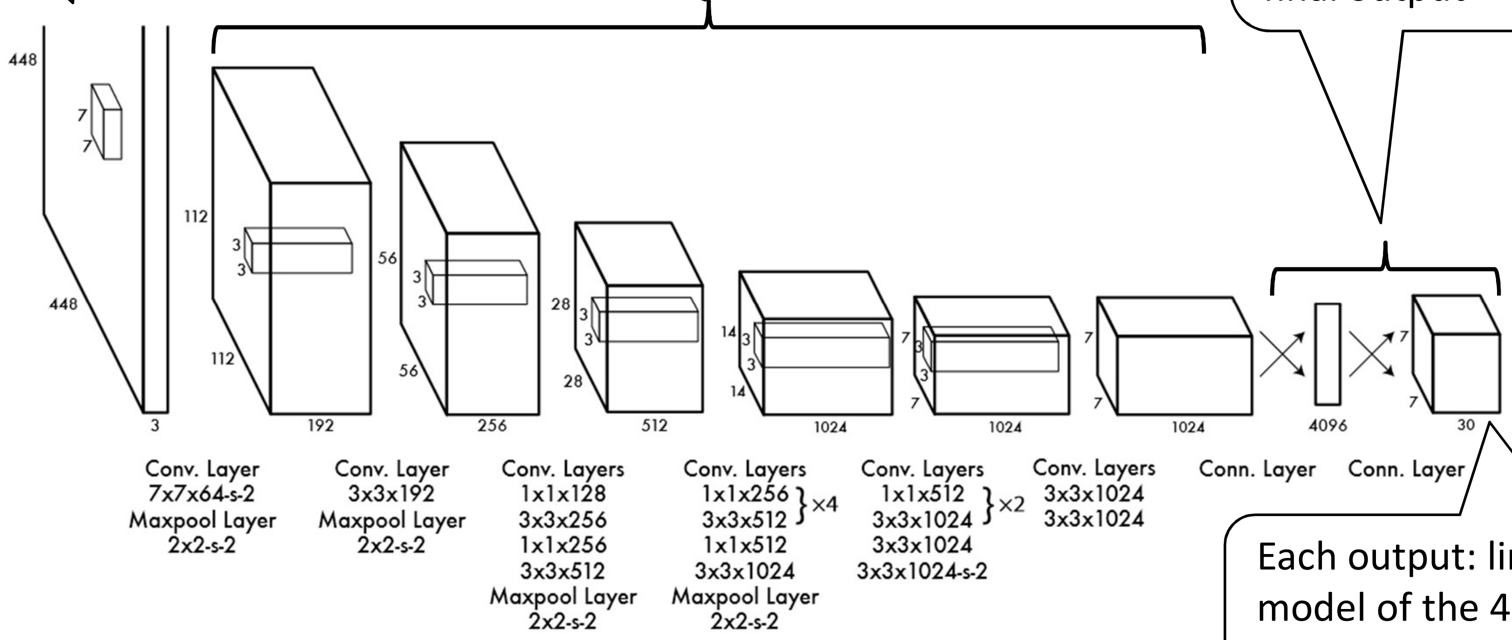
Model Architecture

- Convolutional neural network architecture to map input image to output tensor

Input: RGB-image of size 448 x 448 pixels

Convolution and max-pooling layers to extract meaningful visual features from input image: can be pre-trained on ImageNet

Fully connected layer (4096 neurons) and fully connected final output

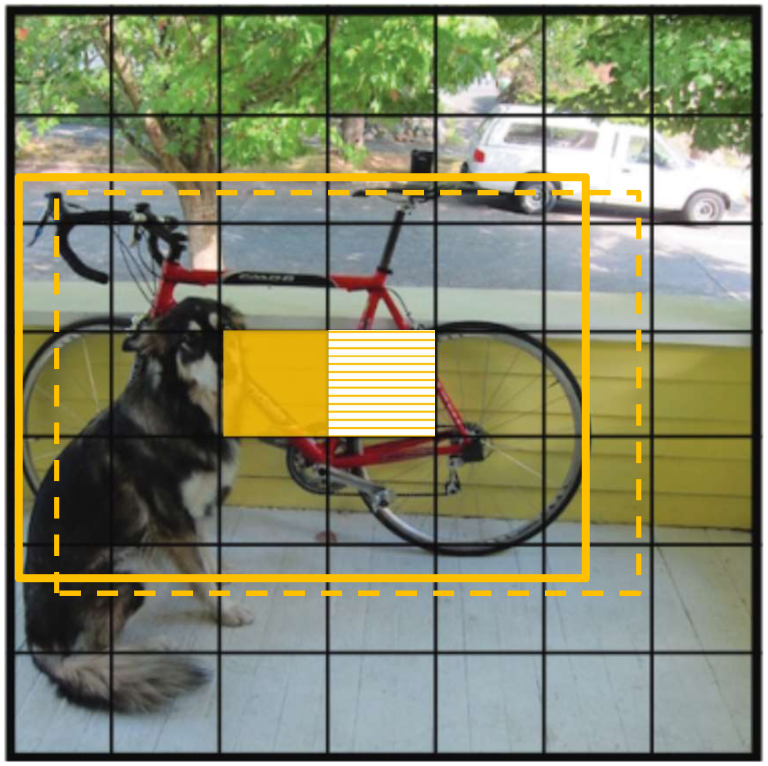


Each output: linear model of the 4096 neurons in last layer

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Non-Max-Suppression

- Non-max-suppression is often applied as a post-processing technique to prevent duplicate detections.

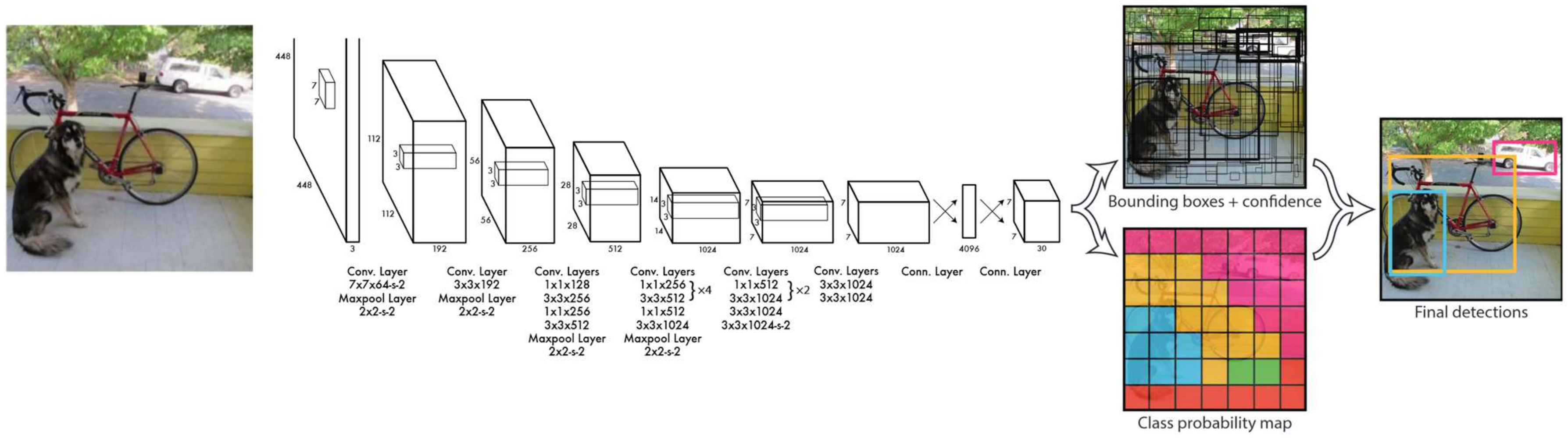


Problem: Neighboring cells sometimes detect the same object with slightly different predicted bounding boxes

Idea: locally only keep the most confident prediction (heuristic, no details)

Overall Pipeline for the YOLO Model

- From input image, model predicts large number of bounding boxes with confidence scores and a map of class probability
- Bounding boxes whose confidence score exceeds a fixed threshold are retained, they are assigned the maximum-probability class of their cell
- Additionally, non-max-suppression can be employed to prevent duplicate detections

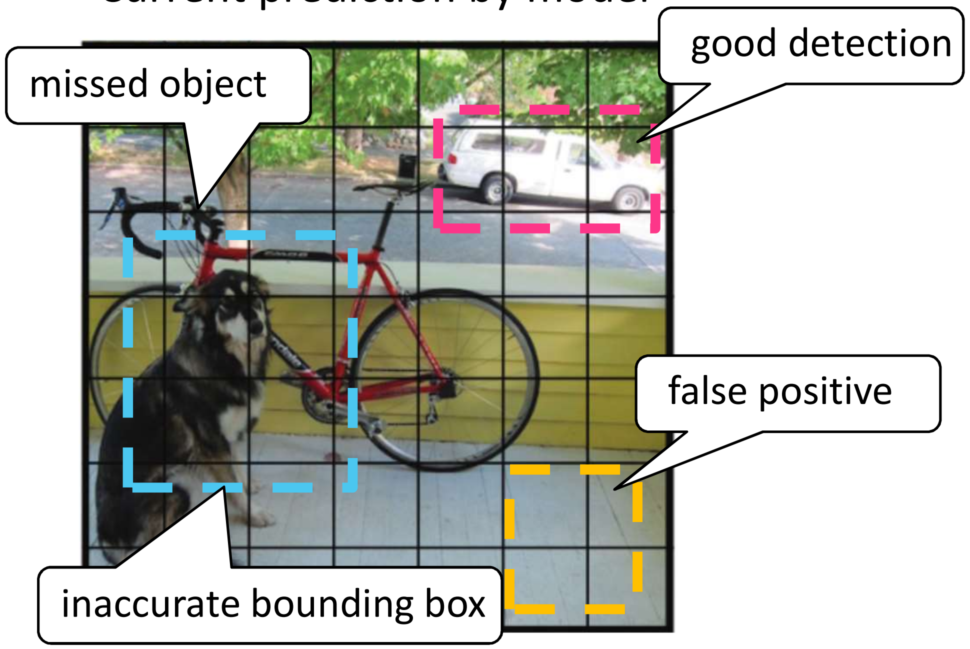


Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

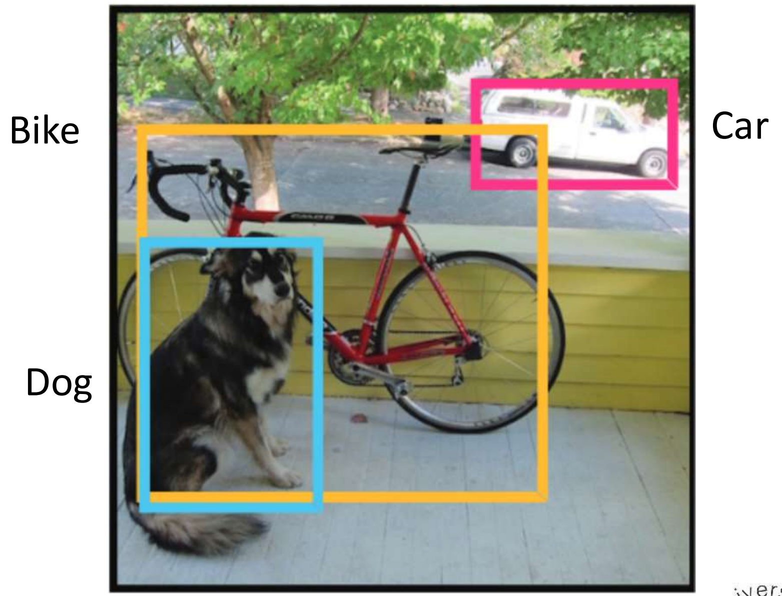
How to Train YOLO?

- Relatively straightforward model and prediction pipeline, but how do we train the weights of the network?
- **Need to define an appropriate loss function**

Current prediction by model

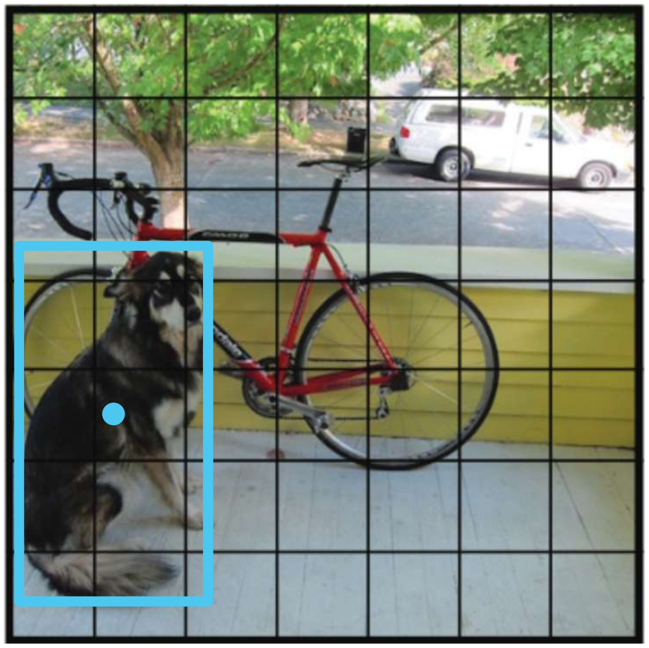


Ground truth (training data): image with a set of ground-truth bounding boxes and classes



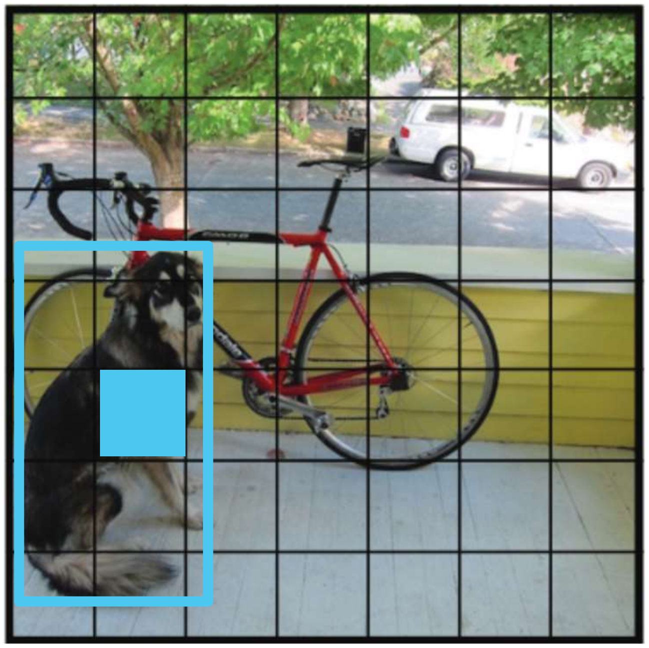
Loss: Match Ground-Truth Annotation to Cell

- Training data: set of ground-truth annotations for each image
- For loss computation, match ground-truth annotation to the right grid cell (by its center)
- Ideally, this cell should predict this bounding box with high confidence and class map should predict the ground-truth class



Loss: Update Class Probabilities for Cell

- Training data: set of ground-truth annotations for each image
- For loss computation, match ground-truth annotation to the right grid cell (by its center)
- Ideally, this cell should predict this bounding box with high confidence and class map should predict the ground-truth class

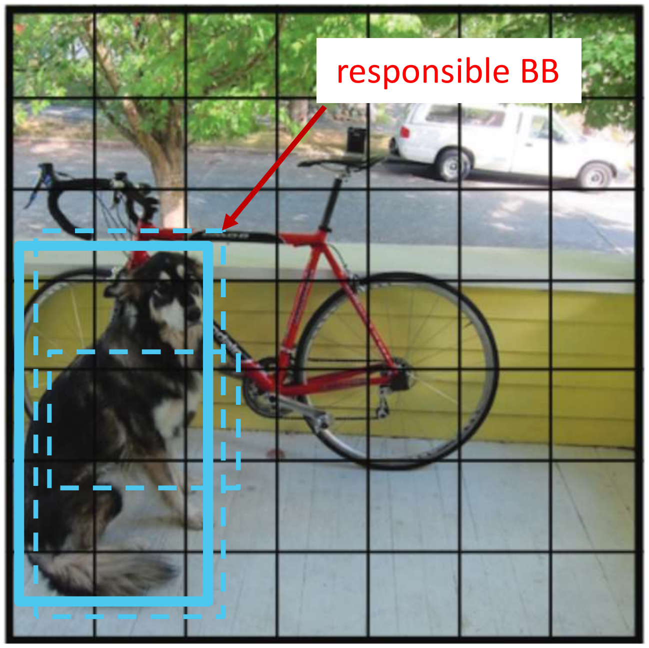


Matching ground-truth annotation to the right cell gives us ground-truth class for cell

Loss function should push class probabilities for cell towards this ground-truth class (details below)

Loss: Update Bounding Boxes

- Loss function should furthermore push predicted bounding boxes towards the bounding boxes in the ground-truth annotations
- Model predicts *B* bounding boxes per cell (often *B*=2). We compute IoU of ground truth bounding box with all (both) predicted bounding boxes. The box with highest IoU is called „responsible“ for the ground truth bounding box.



For responsible bounding box:

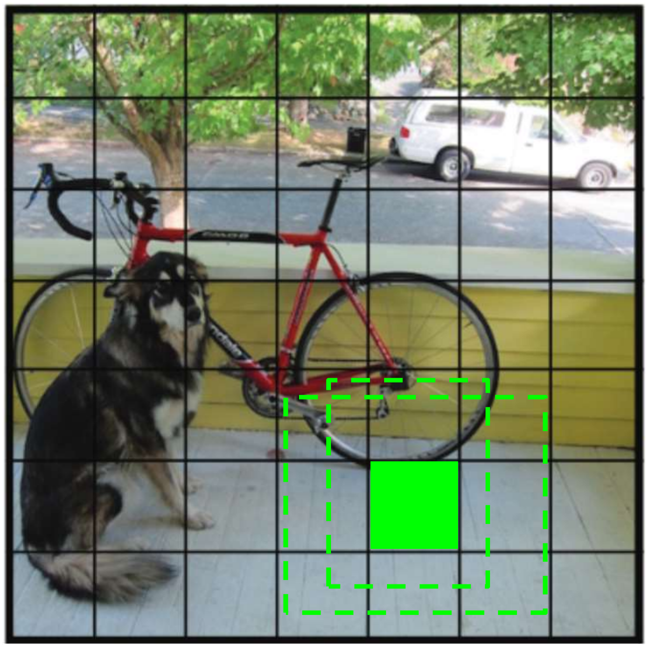
- **Loss function should push *x,y,w,h* towards ground-truth values.**
- **Loss function should adjust confidence: push towards IoU(prediction,ground truth).**

For other bounding boxes:

- **Loss function should decrease confidence (not the right detection for this cell)**
- *x,y,w,h* values remain untouched

Loss: False-positive Predictions

- For many cells, there will be no ground-truth object
- However, *B* bounding boxes are predicted for each cell
- Their confidence should be pushed towards zero by loss function



Predictions without ground-truth object:

- **Loss function should decrease confidence (no object here)**
- *x,y,w,h* values remain untouched
- class probabilities for cell remain untouched

Formal Loss Function of YOLO

- Overall loss for image **x** with ground truth annotations ***y:***

$$
\begin{aligned}
\ell(f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) = {} & \left.\begin{aligned} & \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] \end{aligned}\right\} \text{Adjust bounding boxes for „responsible“ bounding boxes} \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \quad \Big\} \text{Adjust confidence for „responsible“ bounding boxes} \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \quad \Big\} \text{Decrease confidence for other bounding boxes} \\
& + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad \Big\} \text{Adjust class probabilities}
\end{aligned}
$$

Weighting factors (hyperparameters):
- balance between penalty on coordinates, confidences, and class labels
- balance between penalizing too high and too low confidence scores (too many negatives overall)

Formal Loss Function of YOLO

- Loss on object center *x,y:*

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

sum over cells

sum over bounding boxes in cell

Indicator: one if there is an object whose center is in cell *i* and the *j*-th bounding box is responsible for it, zero otherwise

ground-truth center x-coordinate of object in annotation

predicted center x-coordinate of object

- => for the „responsible“ bounding box (the one with highest IoU with ground truth annotation), adjust the center x/y coordinates towards the ground truth

Formal Loss Function of YOLO

- Loss on object size *w,h:*

$$\ell(f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{y}) = \lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \Big[(x$$

squared root: penalizes deviations more for smaller objects than for larger objects

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

ground-truth width of object in annotation

predicted width of object in annotation

- => for the „responsible“ bounding box (the one with highest IoU with ground truth annotation), adjust the width/height towards the ground truth

Formal Loss Function of YOLO

- Loss on confidence for respondible bounding box:

$$\ell(f_\theta(\mathbf{x}), \mathbf{y}) = \lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] + \lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

Ground truth confidence: defined as IoU(predicted bounding box, ground truth bounding box)

predicted confidence for object

- => for the „responsible“ bounding box, adjust the confidence towards IoU of currently predicted and ground truth bounding box. Rationale: want high confidences only for predictions where bounding box is good.

Formal Loss Function of YOLO

- Loss on confidence for respondible bounding box:

$$\ell \quad {}^{\text{obj}}_{ij}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right]$$

Indicator: one if there is no object in cell or there is an object but bounding box prediction *j* is not respondible (not the best match). Zero otherwise.

$$\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\Bigg]$$

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

predicted confidence for bounding box

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

zero

- => for the remaining bounding boxes (which are predictions we do not want), confidence is lowered.

Formal Loss Function of YOLO

- Loss on confidence for respondible bounding box:

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

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

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

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

Indicator: one if there is object with center within cell, zero otherwise

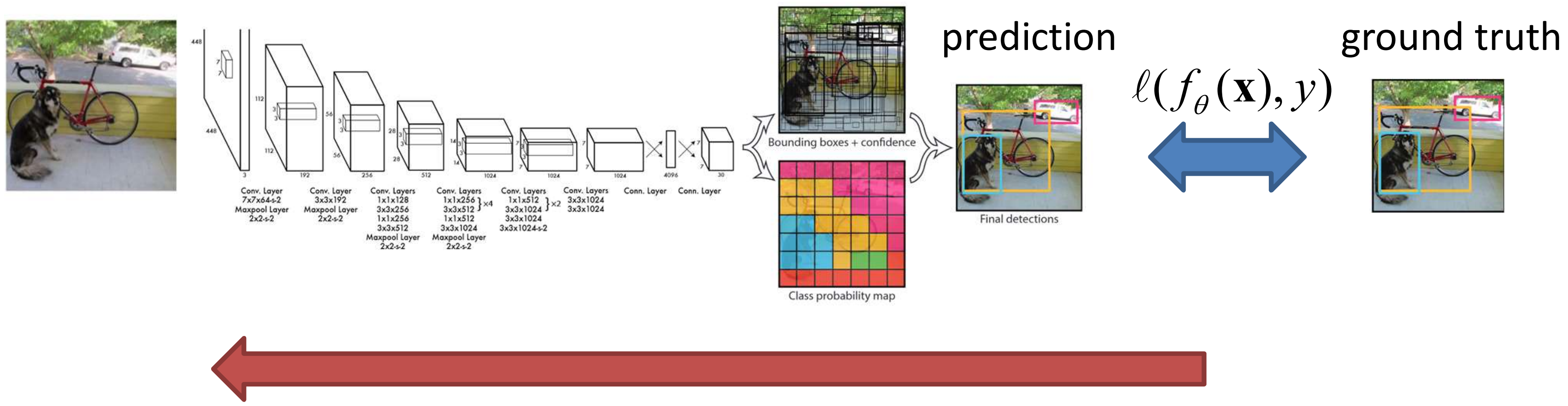
one for the true class, zero otherwise

predicted class probabilities

- When there is an object centered in a cell, adjust the class probabilities for that cell towards the true class of the object

End-to-end Training by SGD

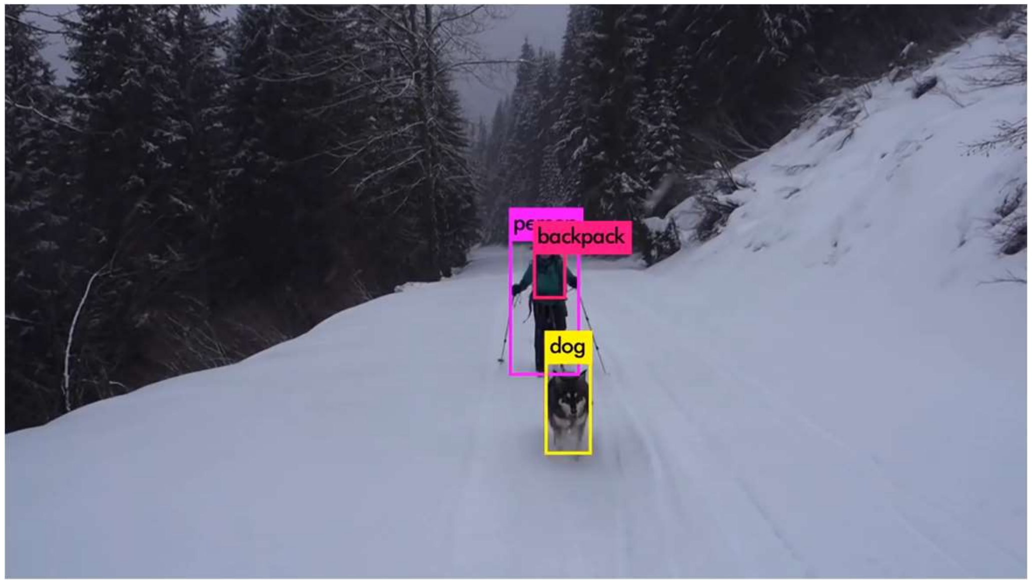
- Given the loss function, model is end-to-end trainable by stochastic gradient descent



Training (parameter estimation)

Computational Performance

- As the YOLO model requires only a single pass through a convolutional neural network architectecture, computational performance is relatively good
- Runs at 45 frames per second on a (fast) standard GPU
- Example:



Demo video of YOLOv3 (slightly improved version of the YOLOv1 model we discussed)

Summary: Object Detection

- Object detection problem setting: recognize several objects on a single image, with bounding box localization and class label

- YOLO architecture for object detection
  - predict a fixed number of bounding boxes for each grid cell within image
  - predict confidence for each bounding box, keep only those bounding boxed whose confidence exceeds threshold
  - loss function to penalize false-positive errors, false-negative errors, misclassifactions, and bounding box inaccuracies
  - fast and reasonably accurate approach