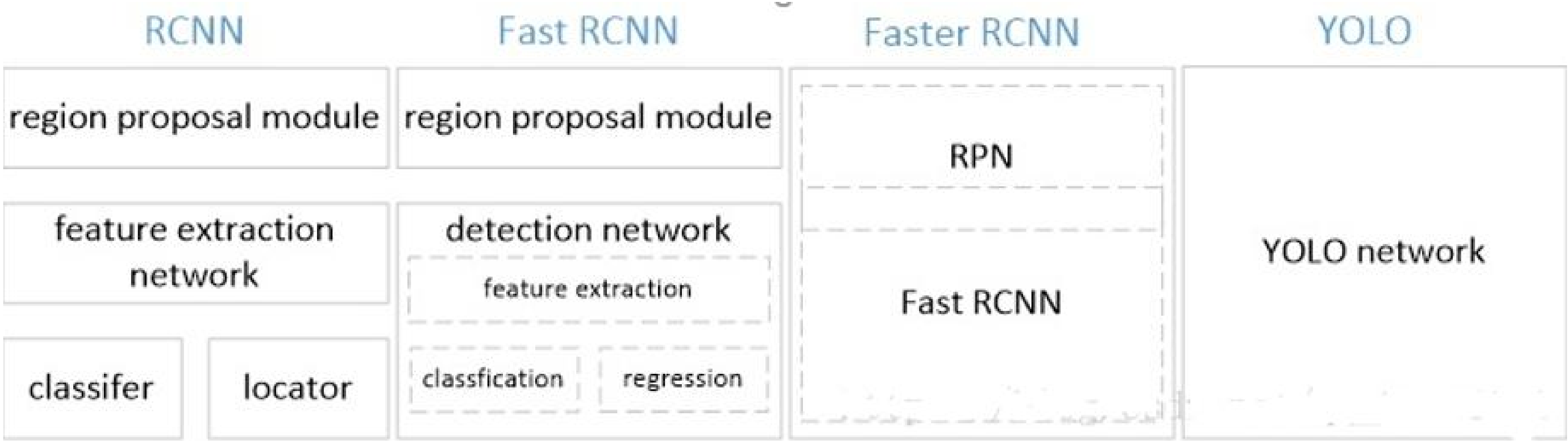
创新

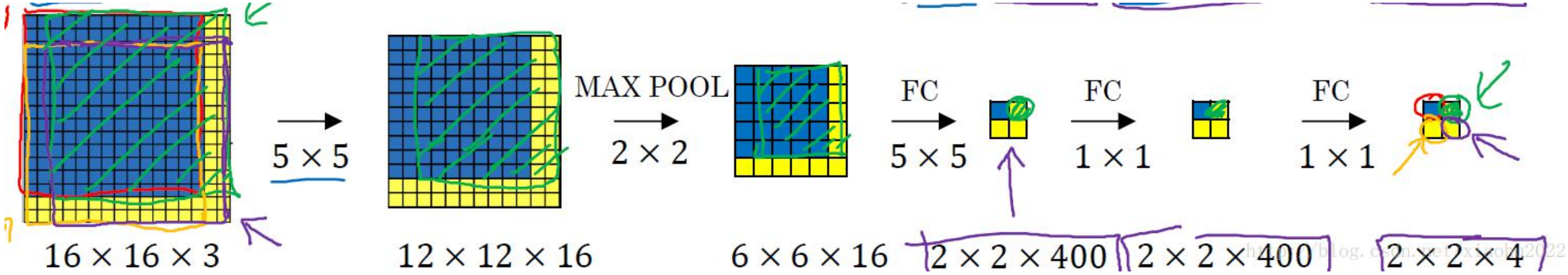


1. YOLO训练和检测均是在一个单独网络中进行
2. YOLO将物体检测作为一个回归问题进行求解

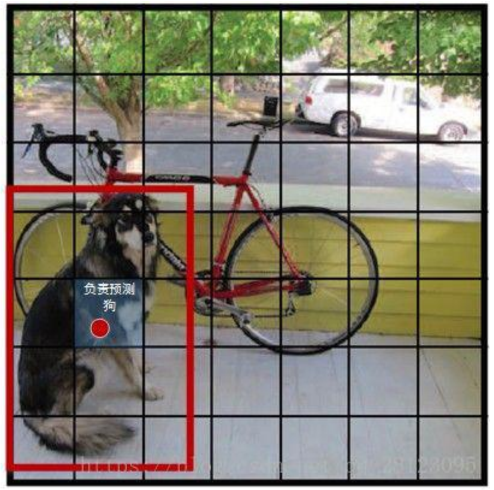
YOLO思想

滑动窗口技术

滑动窗口的CNN实现



YOLO



可以认为特征图的每个元素也是对应原始图片的一个小方块，
然后用每个元素来可以预测那些中心点在该小方格内的目标

YOLO思想

For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.

2. Unified Detection

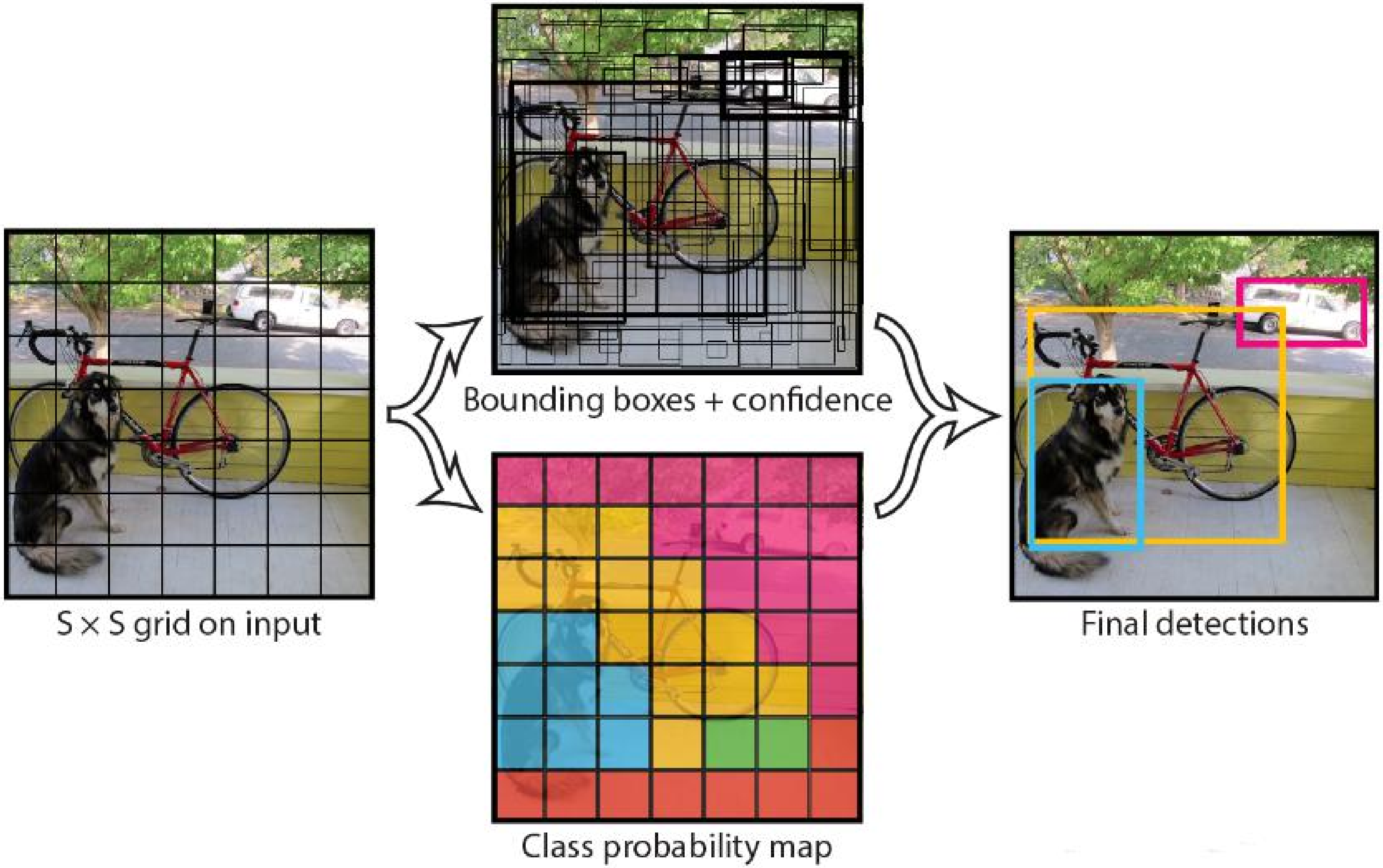
We unify the separate components of object detection into a single neural network. Our network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means our network reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and real-time speeds while maintaining high average precision.

Our system divides the input image into an $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

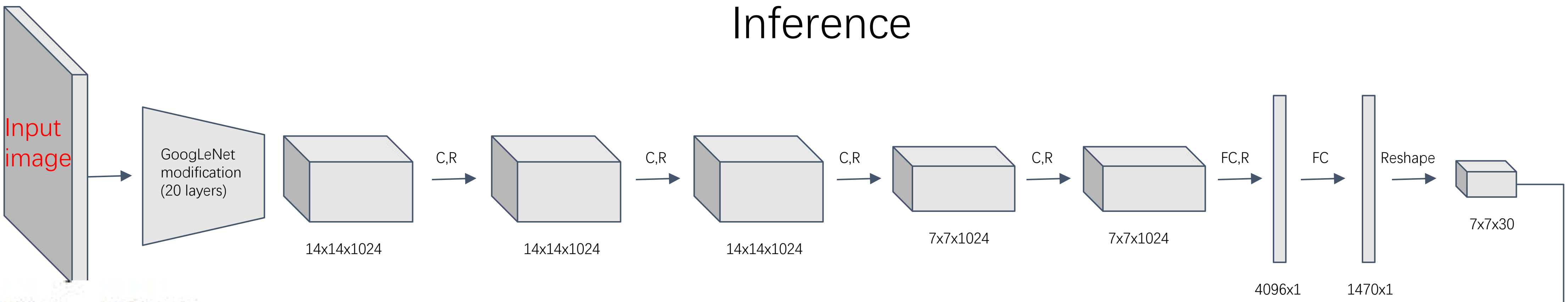
Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as $\Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}}$. If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box.

Each grid cell also predicts C conditional class probabilities, $\Pr(\text{Class}_i|\text{Object})$. These probabilities are conditioned on the grid cell containing an object. We only predict



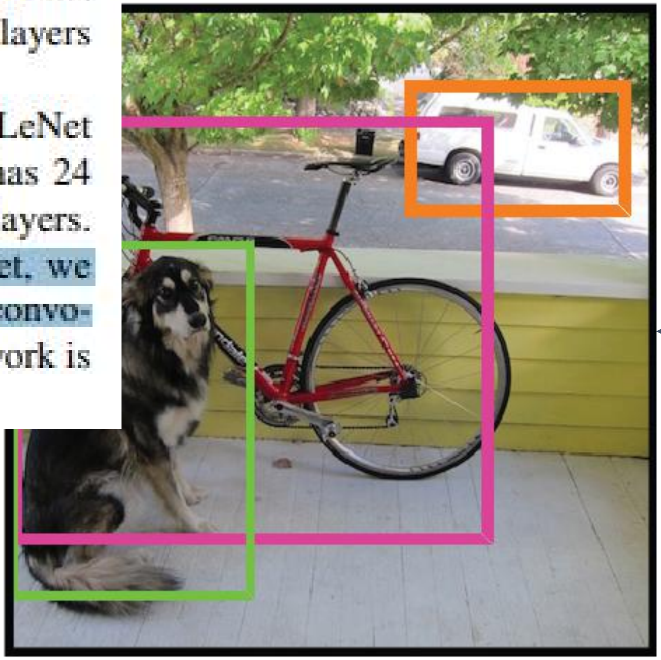
Training



2.1. Network Design

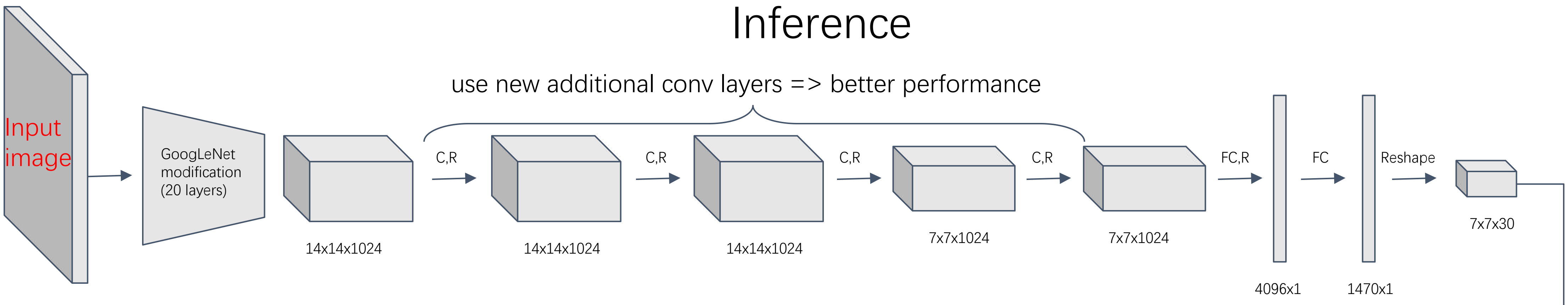
We implement this model as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset [9]. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.

Our network architecture is inspired by the GoogLeNet model for image classification [34]. Our network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1×1 reduction layers followed by 3×3 convolutional layers, similar to Lin et al [22]. The full network is shown in Figure 3.



Detection Procedure

Training

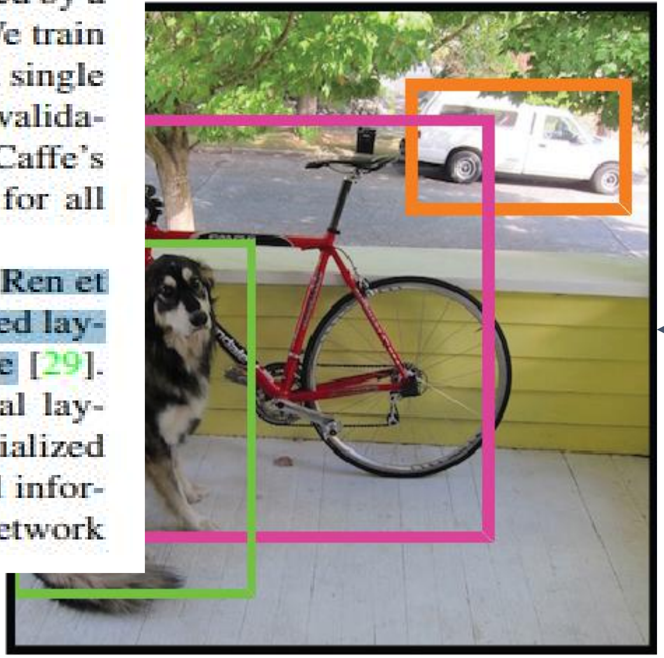


Detection Procedure

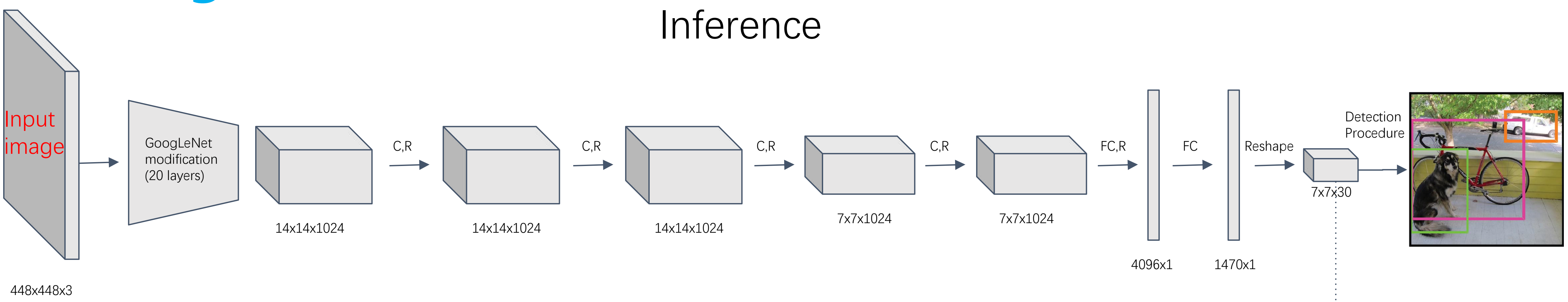
2.2. Training

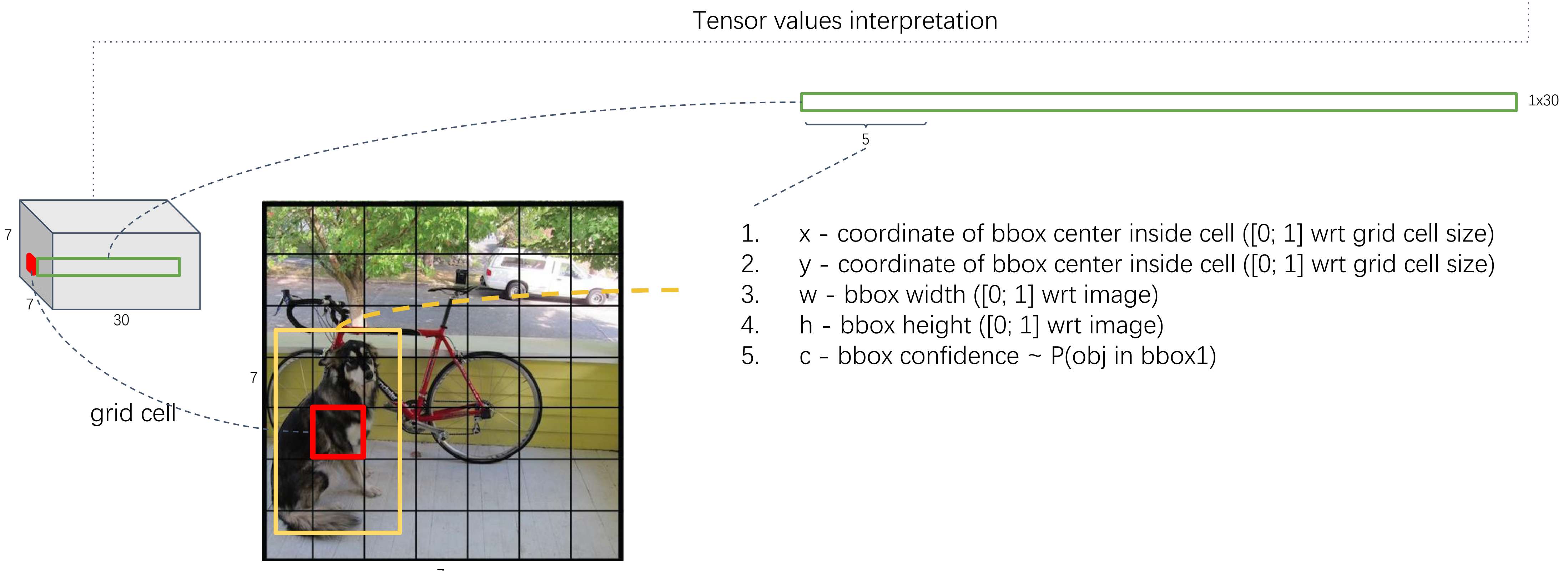
We pretrain our convolutional layers on the ImageNet 1000-class competition dataset [30]. For pretraining we use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. We train this network for approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, comparable to the GoogLeNet models in Caffe's Model Zoo [24]. We use the Darknet framework for all training and inference [26].

We then convert the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can improve performance [29]. Following their example, we add four convolutional layers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual information so we increase the input resolution of the network from 224×224 to 448×448



Training





1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence ~ P(obj in bbox1)

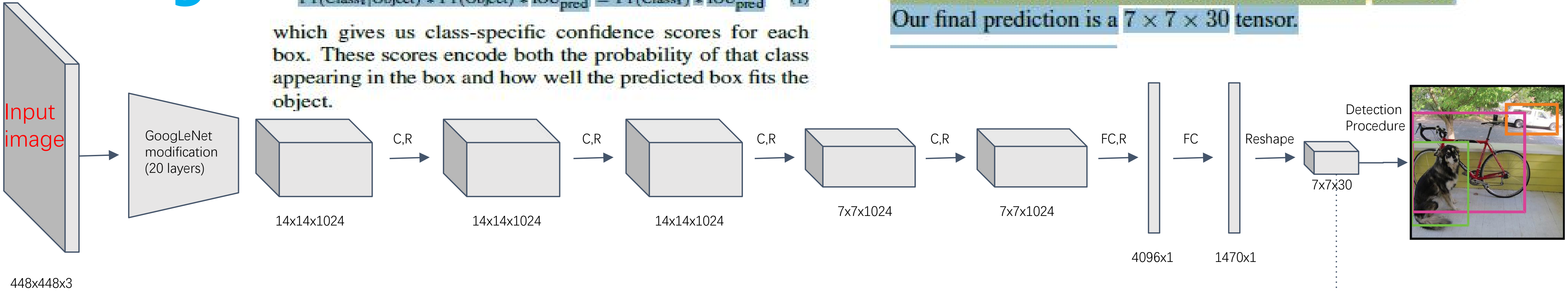
Training

At test time we multiply the conditional class probabilities and the individual box confidence predictions,

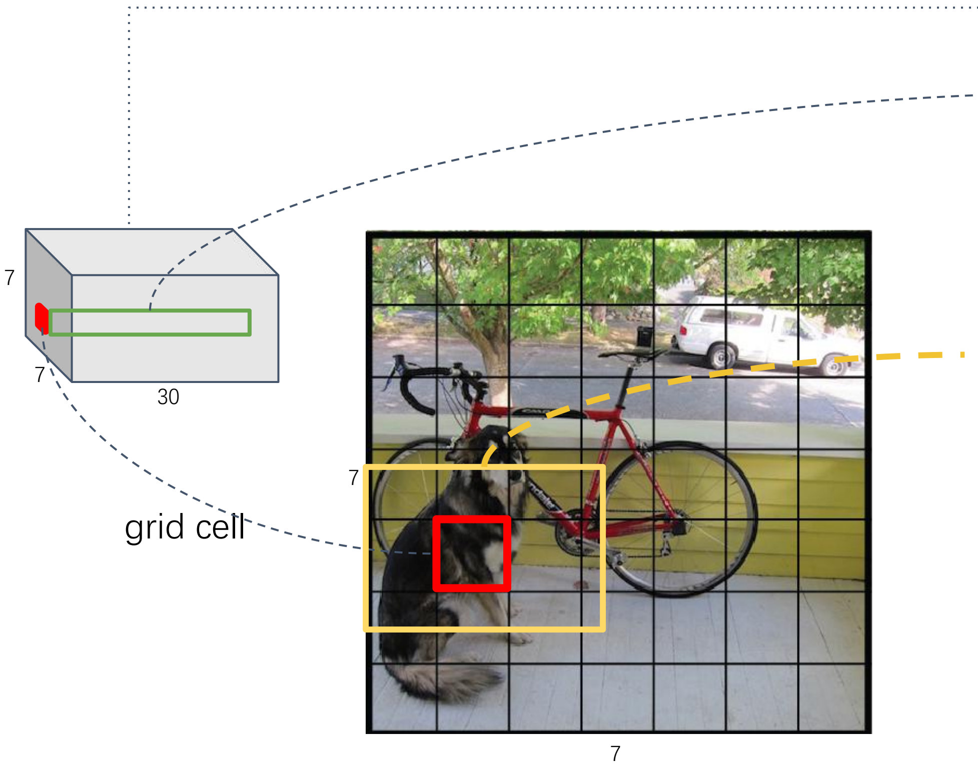
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \quad (1)$$

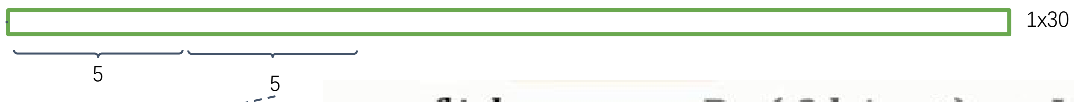
which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.



Tensor values interpretation





$$confidence = Pr(Object) \times IoU_{pred}^{truth}$$

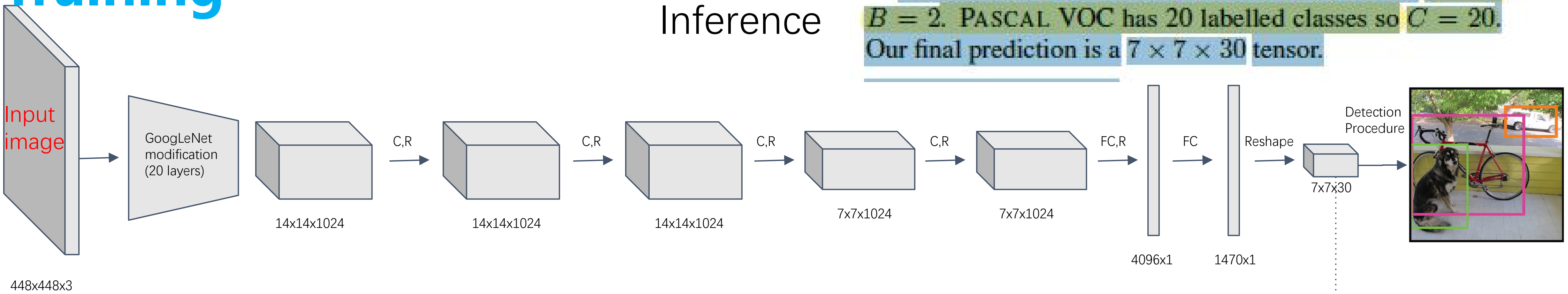
1. x - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
2. y - coordinate of bbox center inside cell ([0; 1] wrt grid cell size)
3. w - bbox width ([0; 1] wrt image)
4. h - bbox height ([0; 1] wrt image)
5. c - bbox confidence

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$. If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

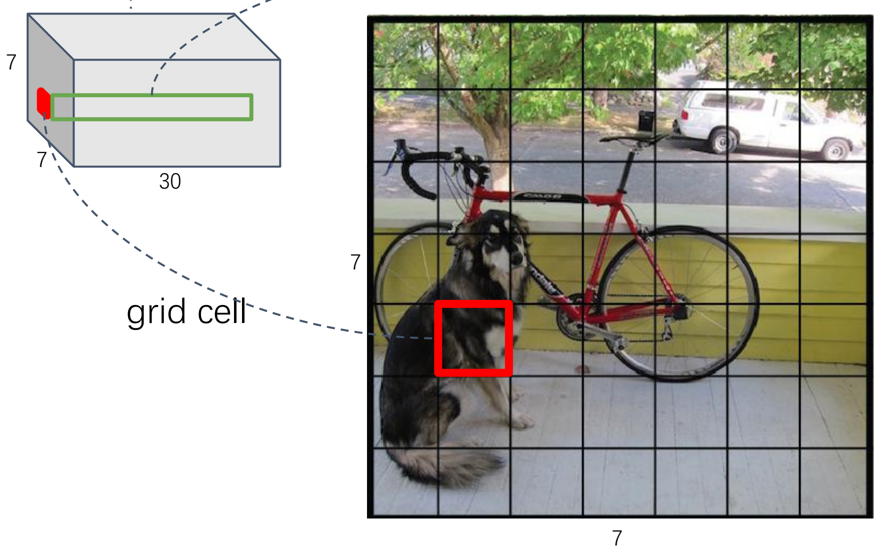
Training

Inference

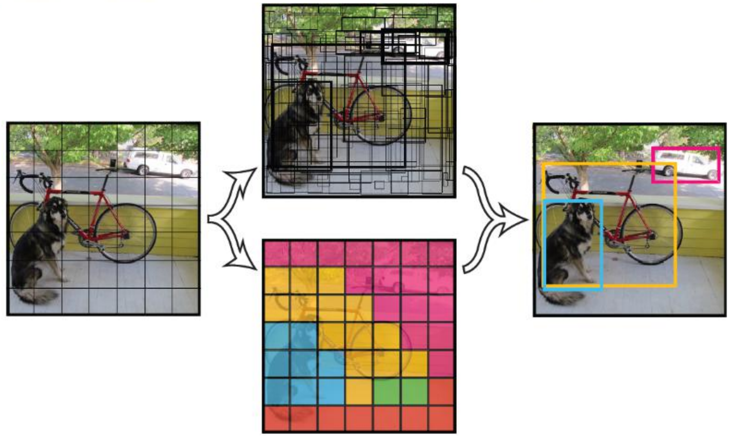
For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.



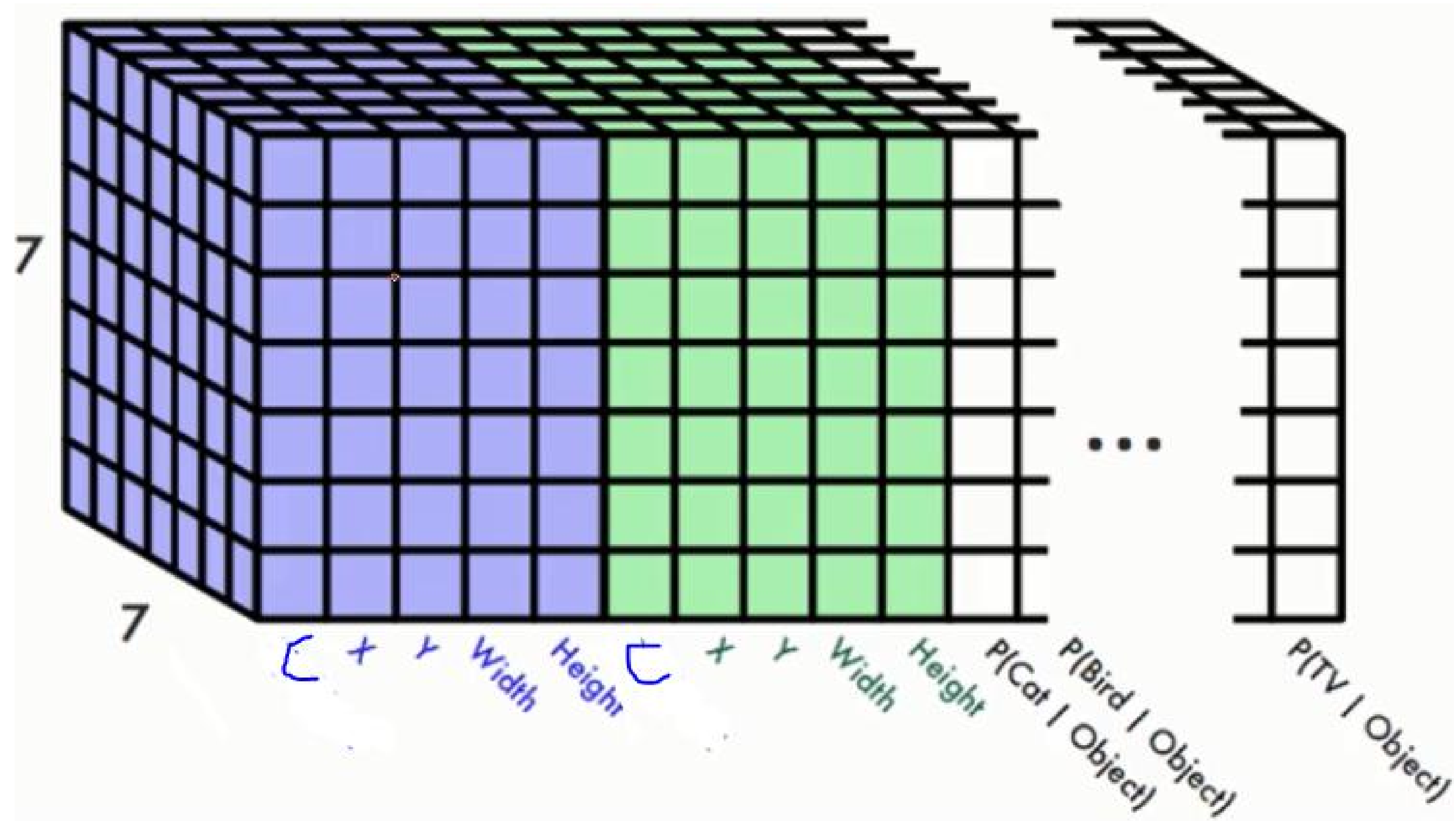
Tensor values interpretation



tioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

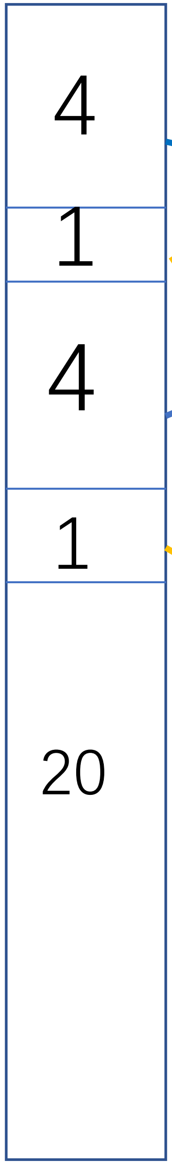


Training



7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

Training Loss



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

坐标预测

判断第i个网格中的第j个box是否负责这个object

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

含object的box的confidence预测

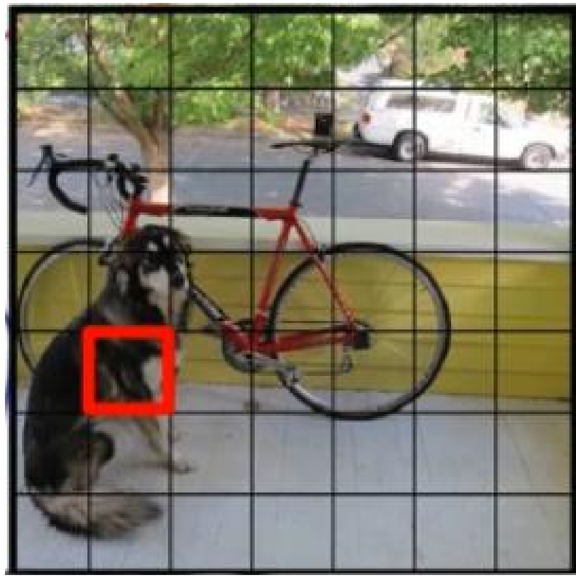
$$+ \lambda_{\mathrm{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\mathrm{noobj}} \left(C_i - \hat{C}_i \right)^2$$

不含object的box的confidence预测

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

类别预测

判断是否有object中心落在网格i中



Training Loss

model. We use sum-squared error because it is easy to optimize, however it does not perfectly align with our goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on.

To remedy this, we increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects. We use two parameters, λ_{coord} and λ_{noobj} to accomplish this. We set $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = .5$.

Sum-squared error also equally weights errors in large boxes and small boxes. Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.

YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.

loss function:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)
\end{aligned}
$$

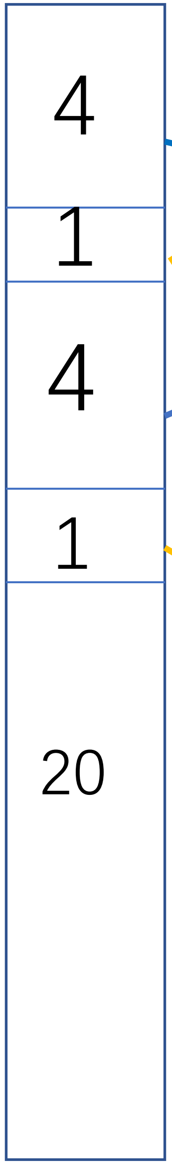
where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

Training Loss

YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor

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.



坐标误差

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

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

坐标预测

判断第i个网格中的第j个box是否负责这个object

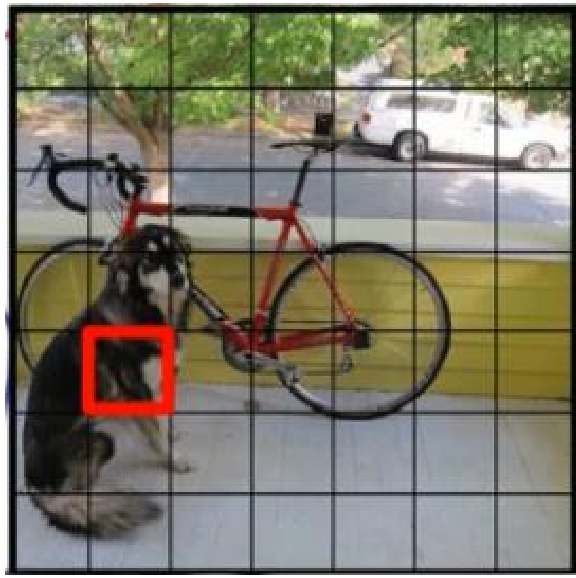
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

含object的box的confidence预测

IOU误差

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

不含object的box的confidence预测



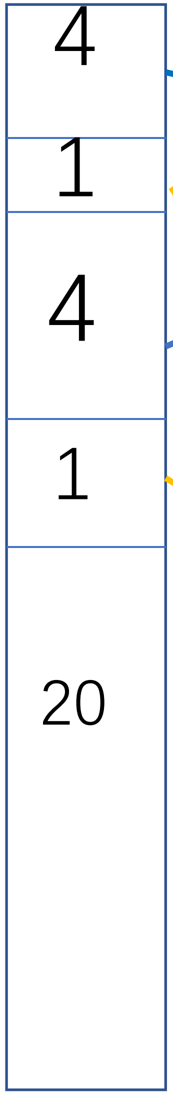
判断是否有object中心落在网格i中

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

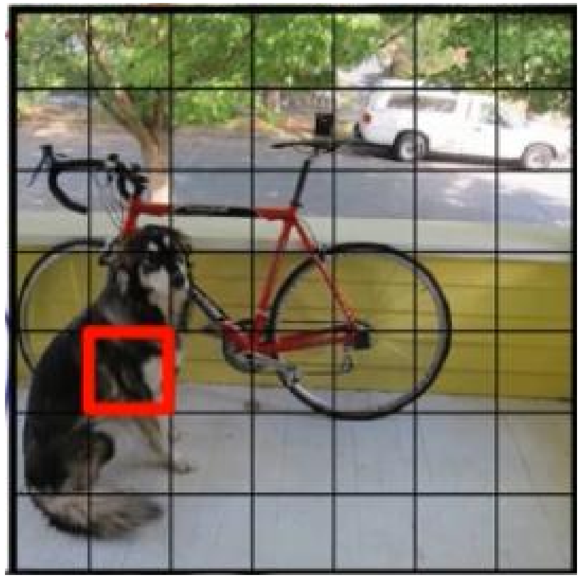
类别预测

分类误差

Training Loss



Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).



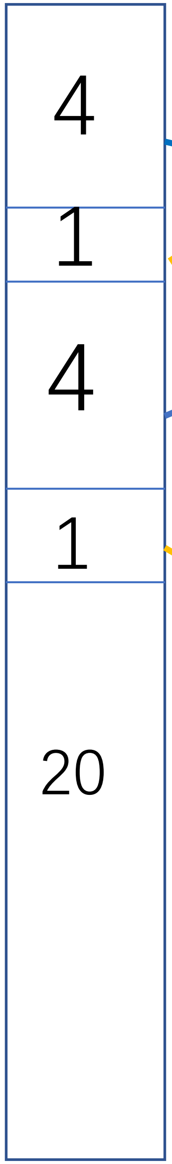
$$+\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

判断是否有object中心落在网格i中

类别预测

分类误差

Training Loss



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

坐标误差

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

坐标预测

判断第i个网格中的第j个box是否负责这个object

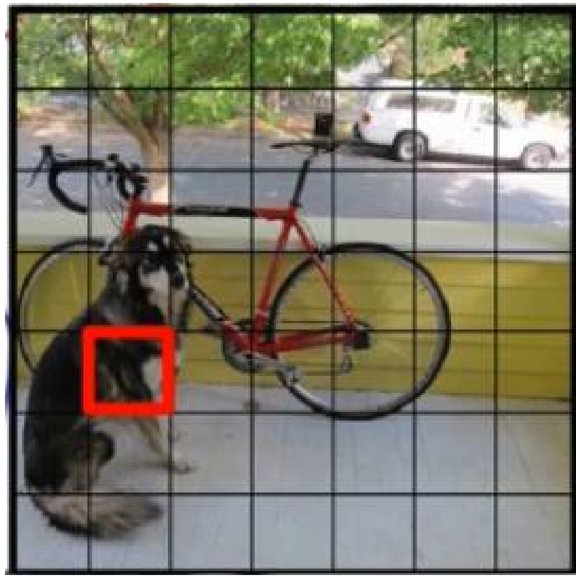
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

含object的box的confidence预测

IOU误差

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

别预测



Training Loss

YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor

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.

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

4

1

4

1

20

坐标误差

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

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

坐标预测

判断第i个网格中的第j个box是否负责这个object

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

含object的box的confidence预测

IOU误差

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

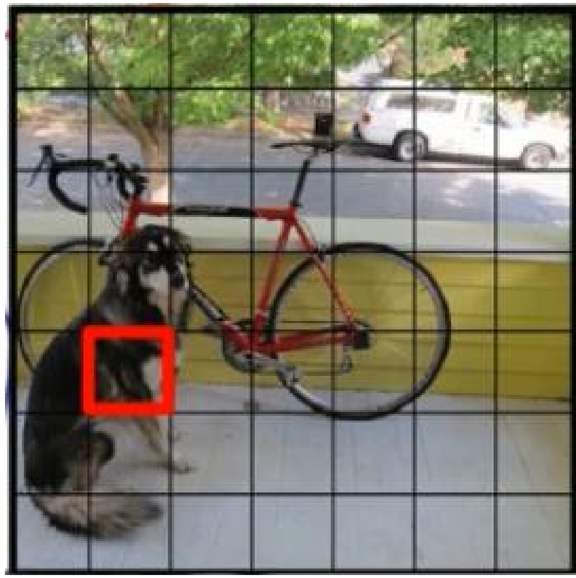
不含object的box的confidence预测

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

判断是否有object中心落在网格i中

类别预测

分类误差



```
if(state.train){  1
    float avg_iou = 0;
    float avg_cat = 0;
    float avg_allcat = 0;
    float avg_obj = 0;
    float avg_anyobj = 0;
    int count = 0;
    *(l.cost) = 0;
    int size = l.inputs * l.batch;
    memset(l.delta, 0, size * sizeof(float));  2
    for (b = 0; b < l.batch; ++b){
        int index = b*l.inputs;  3
        for (i = 0; i < locations; ++i) {  4
            int truth_index = (b*locations + i)*(1+l.coords+l.classes);  5
            int is_obj = state.truth[truth_index];  6
            for (j = 0; j < l.n; ++j) {  7
                int p_index = index + locations*l.classes + i*l.n + j;
                l.delta[p_index] = l.noobject_scale*(0 - l.output[p_index]);
                *(l.cost) += l.noobject_scale*pow(l.output[p_index], 2);  8
                avg_anyobj += l.output[p_index];
            }

            int best_index = -1;
            float best_iou = 0;
            float best_rmse = 20;

            if (!is_obj){
                continue;  9
            }

            int class_index = index + i*l.classes;  10
            for(j = 0; j < l.classes; ++j) {
                l.delta[class_index+j] = l.class_scale * (state.truth[truth_index+1+j] - l.output[class_index+j]);
                *(l.cost) += l.class_scale * pow(state.truth[truth_index+1+j] - l.output[class_index+j], 2);
                if(state.truth[truth_index + 1 + j]) avg_cat += l.output[class_index+j];
                avg_allcat += l.output[class_index+j];
```

坐标误差

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

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

坐标预测

判断第i个网格中的第j个box是否负责这个object

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

含object的box的confidence预测

IOU误差

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

不含object的box的confidence预测

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

类别预测

判断是否有object中心落在网格i中

分类误差

```
    int class_index = index + i*l.classes;
    for(j = 0; j < l.classes; ++j) {
11      l.delta[class_index+j] = l.class_scale * (state.truth[truth_index+1+j] - l.output[class_index+j]);
        *(l.cost) += l.class_scale * pow(state.truth[truth_index+1+j] - l.output[class_index+j], 2);
        if(state.truth[truth_index + 1 + j]) avg_cat += l.output[class_index+j];
        avg_allcat += l.output[class_index+j];
    }

12  box truth = float_to_box(state.truth + truth_index + 1 + l.classes);
    truth.x /= l.side;
    truth.y /= l.side;

    for(j = 0; j < l.n; ++j){   <-- 13
        int box_index = index + locations*(l.classes + l.n) + (i*l.n + j) * l.co
        box out = float_to_box(l.output + box_index);
        out.x /= l.side;
        out.y /= l.side;

        if (l.sqrt){
            out.w = out.w*out.w;
            out.h = out.h*out.h;
        }

        float iou  = box_iou(out, truth);   <-- 14
        //iou = 0;
        float rmse = box_rmse(out, truth);
        if(best_iou > 0 || iou > 0){   <-- 15
            if(iou > best_iou){
                best_iou = iou;
                best_index = j;
            }
        }else{
            if(rmse < best_rmse){
                best_rmse = rmse;
                best_index = j;
```

坐标误差

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

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

坐标预测

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

含object的box的confidence预测

判断第i个网格中的第j个box是否负责这个object

IOU误差

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

不含object的box的confidence预测

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

类别预测

判断是否有object中心落在网格i中

分类误差

```c
if(l.random && *(state.net.seen) < 64000){
```
16
```c
    best_index = rand()%l.n;
}

int box_index = index + locations*(l.classes + l.n) + (i*l.n + best_index) * l.coords;
int tbox_index = truth_index + 1 + l.classes;

box out = float_to_box(l.output + box_index);
out.x /= l.side;
out.y /= l.side;
if (l.sqrt) {
    out.w = out.w*out.w;
    out.h = out.h*out.h;
}
float iou  = box_iou(out, truth);

//printf("%d,", best_index);
int p_index = index + locations*l.classes + i*l.n + best_index;
*(l.cost) -= l.noobject_scale * pow(l.output[p_index], 2);
```
17
```c
*(l.cost) += l.object_scale * pow(1-l.output[p_index], 2);
avg_obj += l.output[p_index];
l.delta[p_index] = l.object_scale * (1.-l.output[p_index]);

if(l.rescore){
    l.delta[p_index] = l.object_scale * (iou - l.output[p_index]);
}
```
18
```c
l.delta[box_index+0] = l.coord_scale*(state.truth[tbox_index + 0] - l.output[box_index + 0]);
l.delta[box_index+1] = l.coord_scale*(state.truth[tbox_index + 1] - l.output[box_index + 1]);
l.delta[box_index+2] = l.coord_scale*(state.truth[tbox_index + 2] - l.output[box_index + 2]);
l.delta[box_index+3] = l.coord_scale*(state.truth[tbox_index + 3] - l.output[box_index + 3]);
if(l.sqrt){
    l.delta[box_index+2] = l.coord_scale*(sqrt(state.truth[tbox_index + 2]) - l.output[box_index + 2]);
    l.delta[box_index+3] = l.coord_scale*(sqrt(state.truth[tbox_index + 3]) - l.output[box_index + 3]);
```

坐标误差

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

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

坐标预测

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

含object的box的confidence预测

判断第i个网格中的第j个box是否负责这个object

IOU误差

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

不含object的box的confidence预测

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

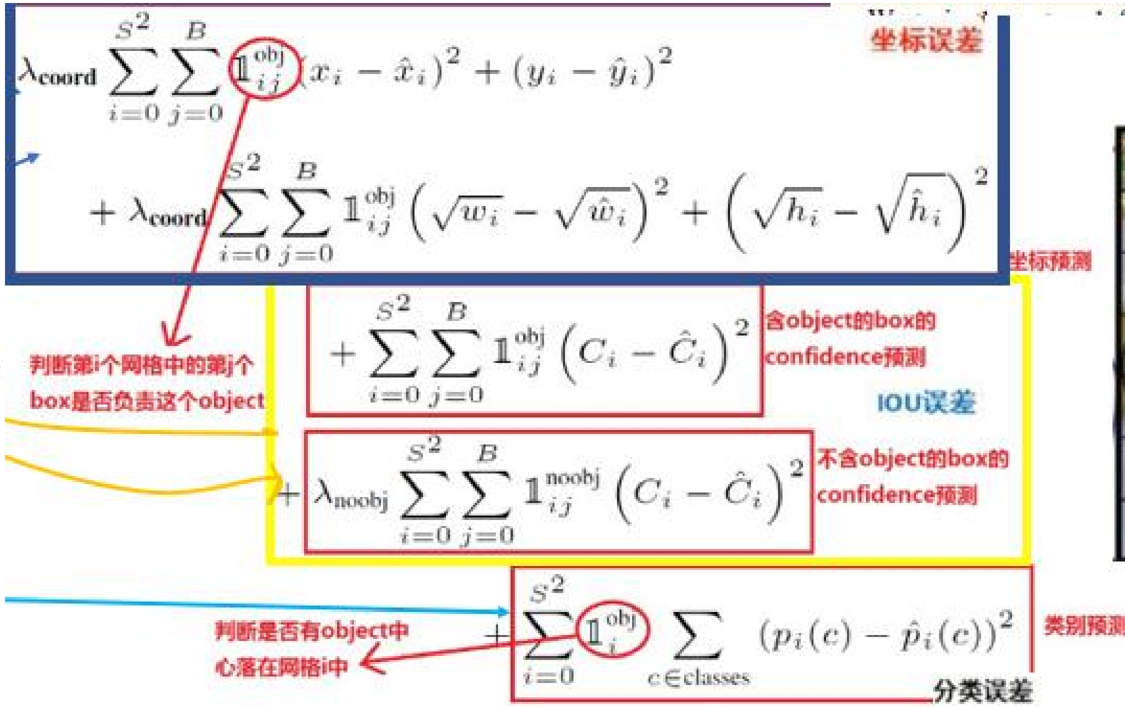
类别预测

判断是否有object中心落在网格i中

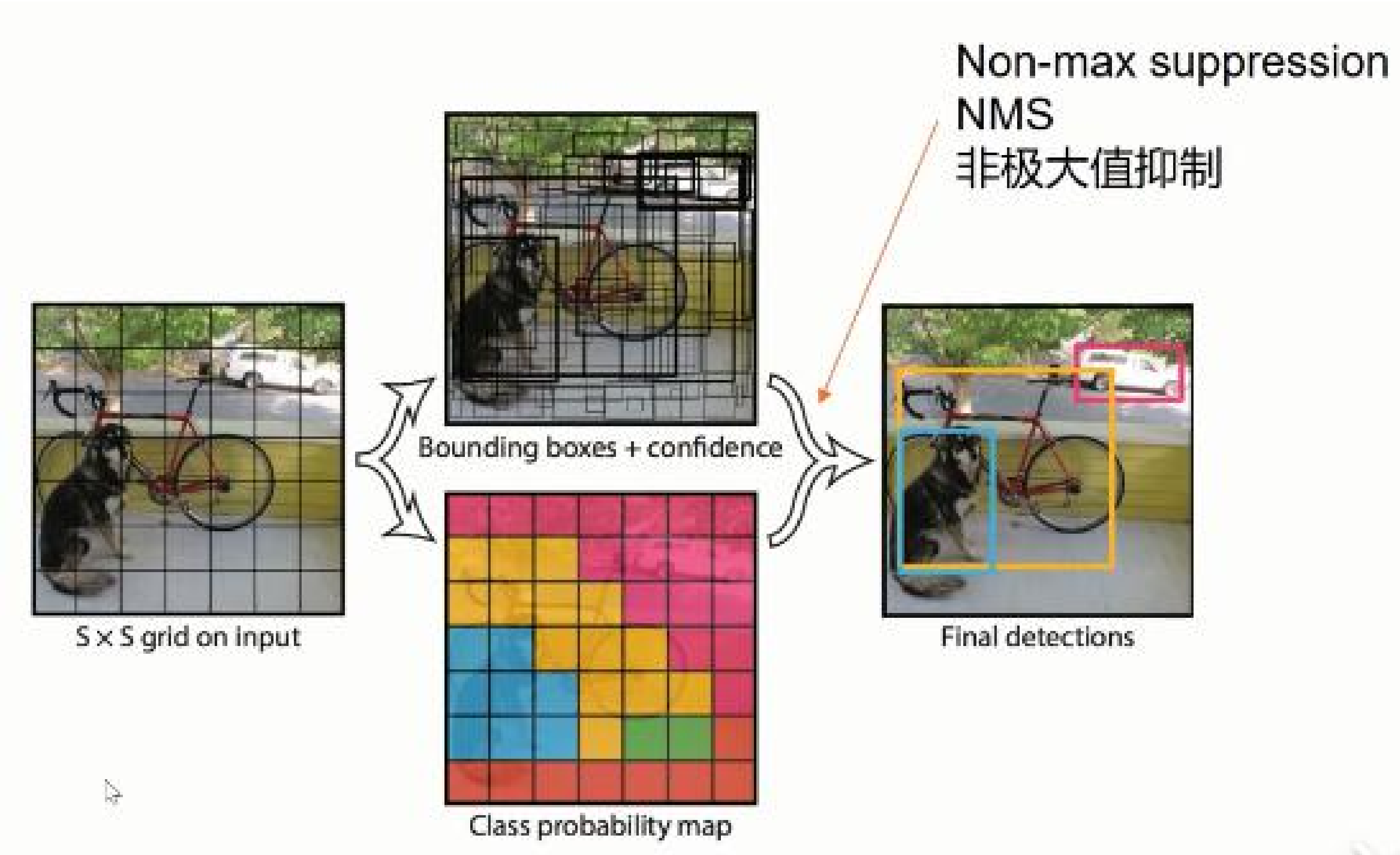
分类误差

```
            *(l.cost) += pow(1-iou, 2);
            avg_iou += iou;
            ++count;
        }
    }

    if(0){
        float *costs = calloc(l.batch*locations*l.n, sizeof(float));
        for (b = 0; b < l.batch; ++b) {
            int index = b*l.inputs;
            for (i = 0; i < locations; ++i) {
                for (j = 0; j < l.n; ++j) {
                    int p_index = index + locations*l.classes + i*l.n + j;
                    costs[b*locations*l.n + i*l.n + j] = l.delta[p_index]*l.delta[p_index];
                }
            }
        }
        int indexes[100];
        top_k(costs, l.batch*locations*l.n, 100, indexes);
        float cutoff = costs[indexes[99]];
        for (b = 0; b < l.batch; ++b) {
            int index = b*l.inputs;
            for (i = 0; i < locations; ++i) {
                for (j = 0; j < l.n; ++j) {
                    int p_index = index + locations*l.classes + i*l.n + j;
                    if (l.delta[p_index]*l.delta[p_index] < cutoff) l.delta[p_index] = 0;
                }
            }
        }
        free(costs);
    }

    *(l.cost) = pow(mag_array(l.delta, l.outputs * l.batch), 2);
```

19

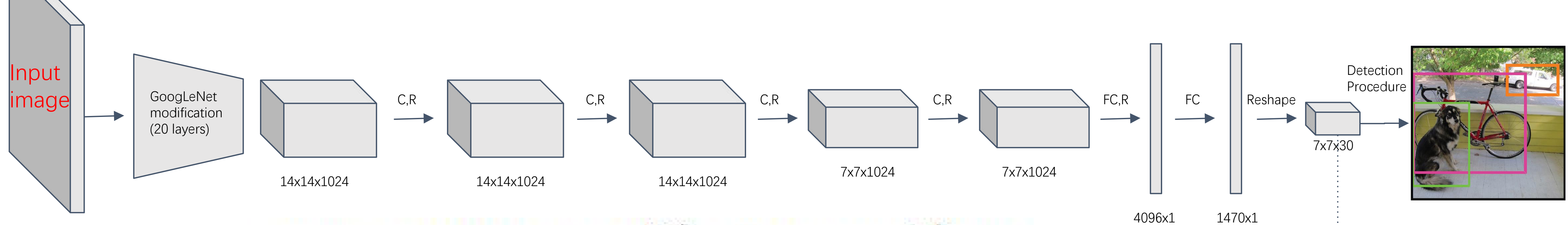


Testing



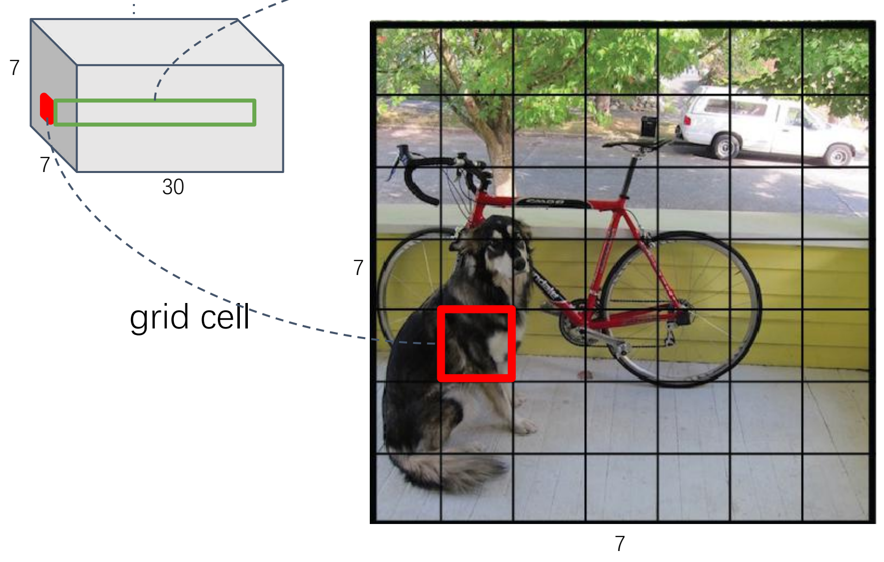
Non-max suppression
NMS
非极大值抑制
Bounding boxes + confidence
S × S grid on input
Final detections
Class probability map

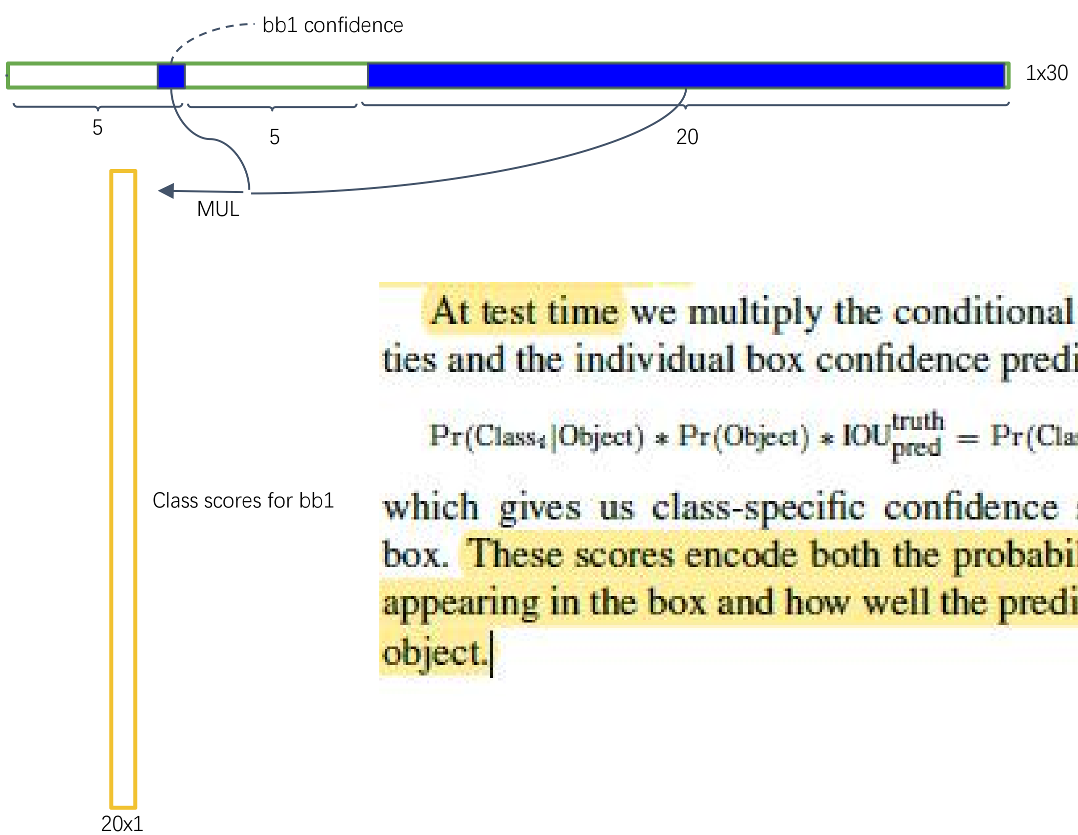
Inference



$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \quad (1)$$

Tensor values interpretation



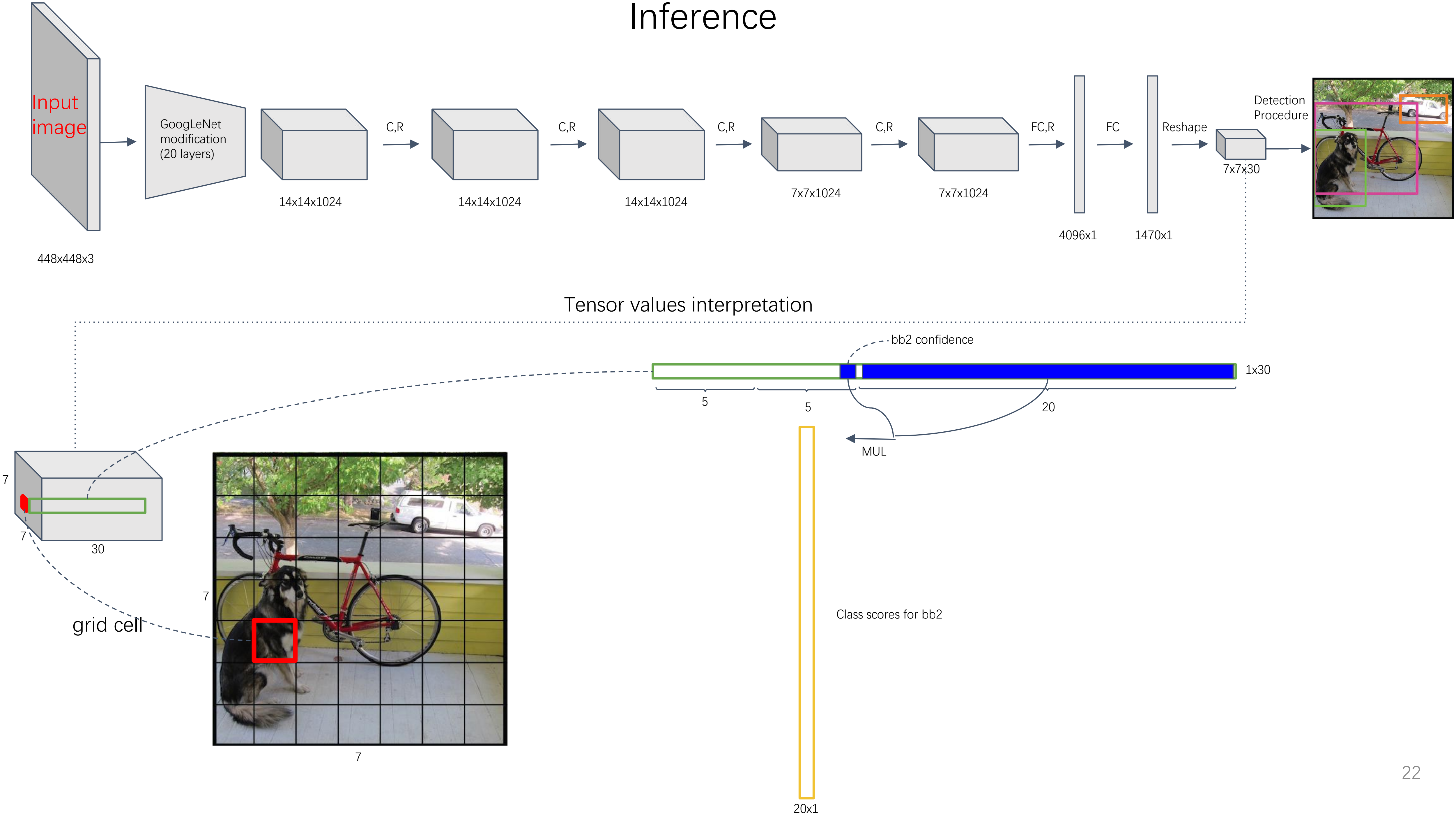


At test time we multiply the conditional class probabilities and the individual box confidence predictions,

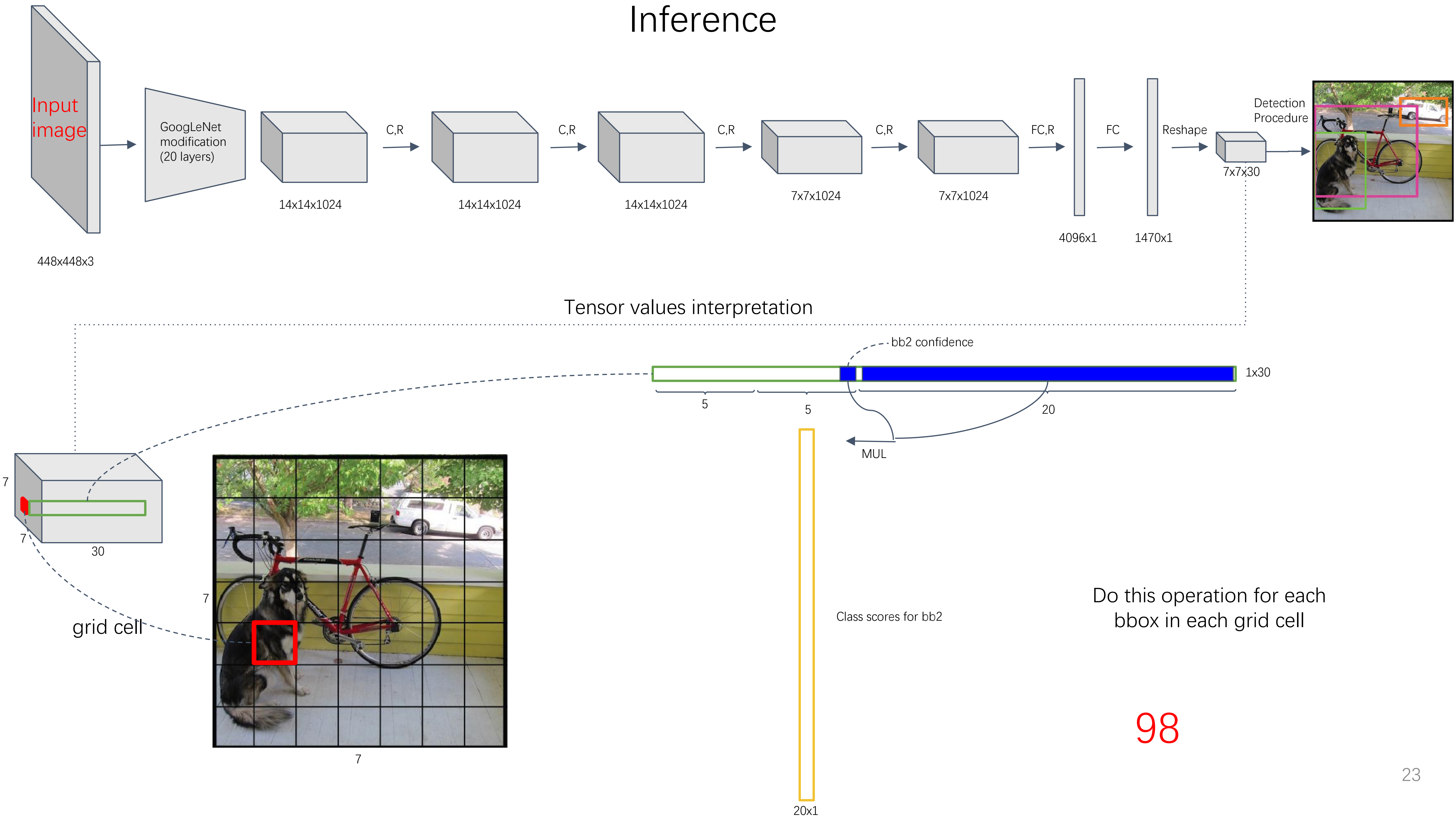
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \quad (1)$$

which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

Inference


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
bb2 confidence
1x30
5
5
20
MUL
Class scores for bb2
20x1
7
7
30
grid cell
7
7

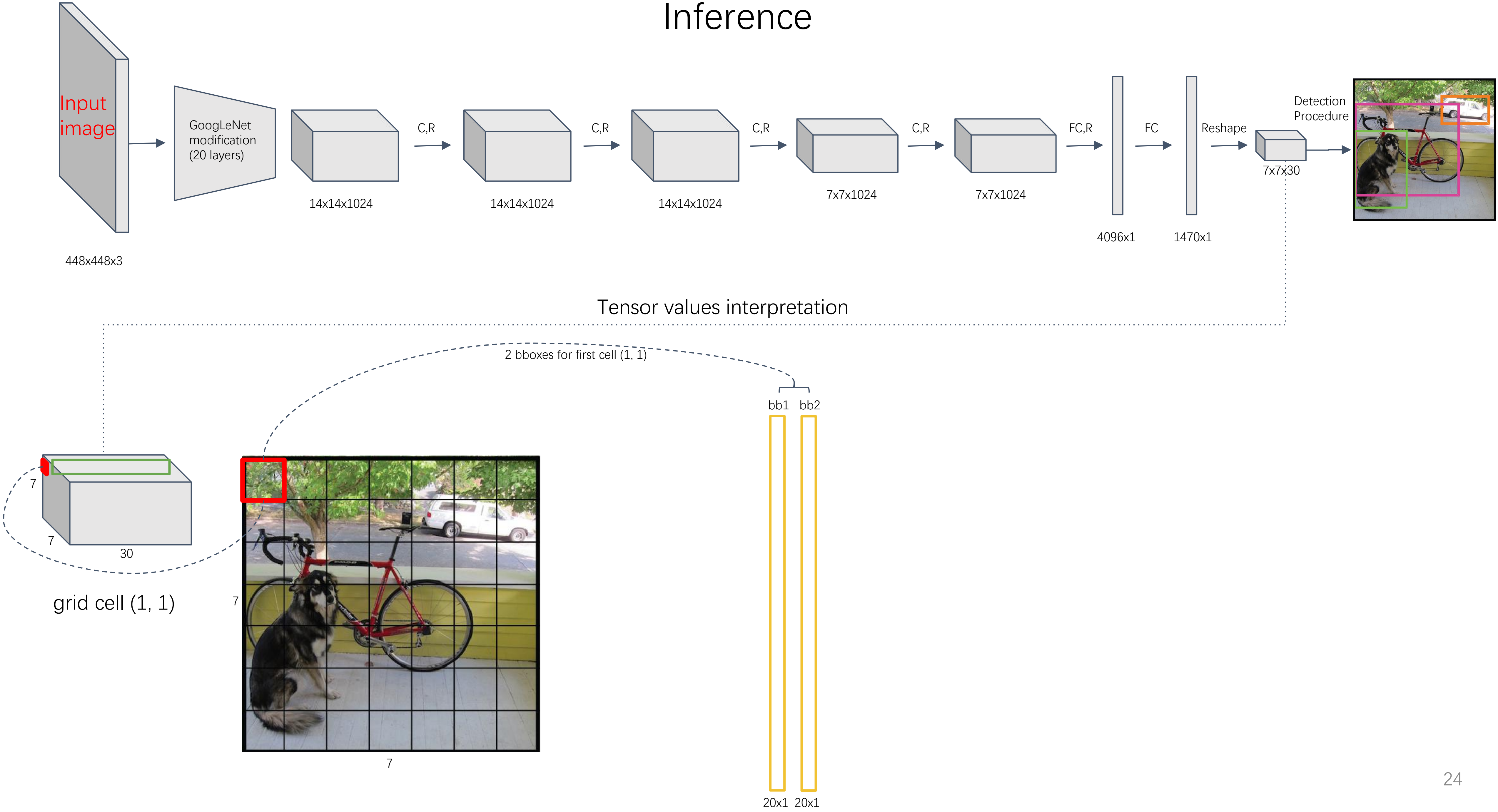
Inference



Do this operation for each bbox in each grid cell

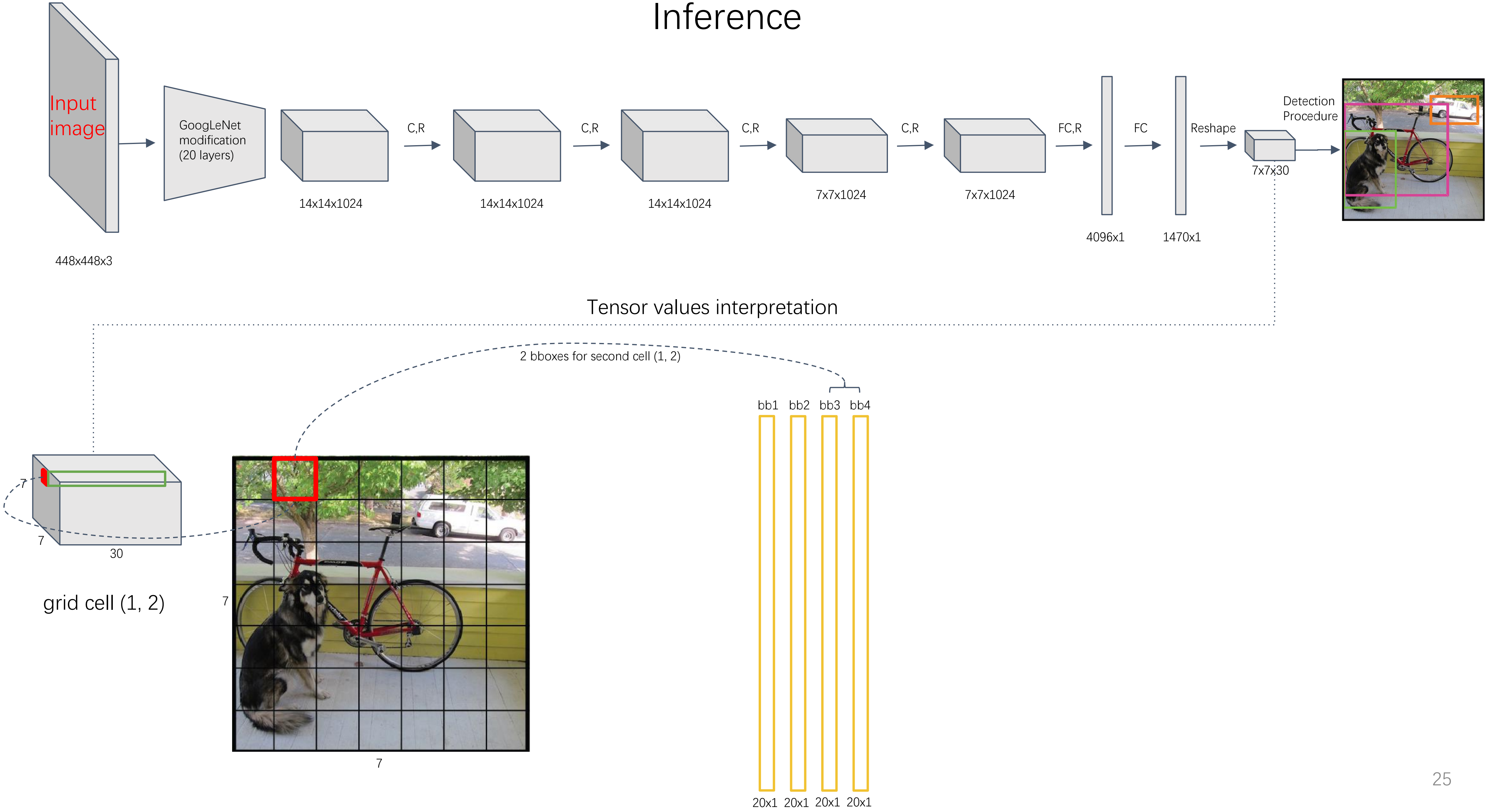
98

Inference

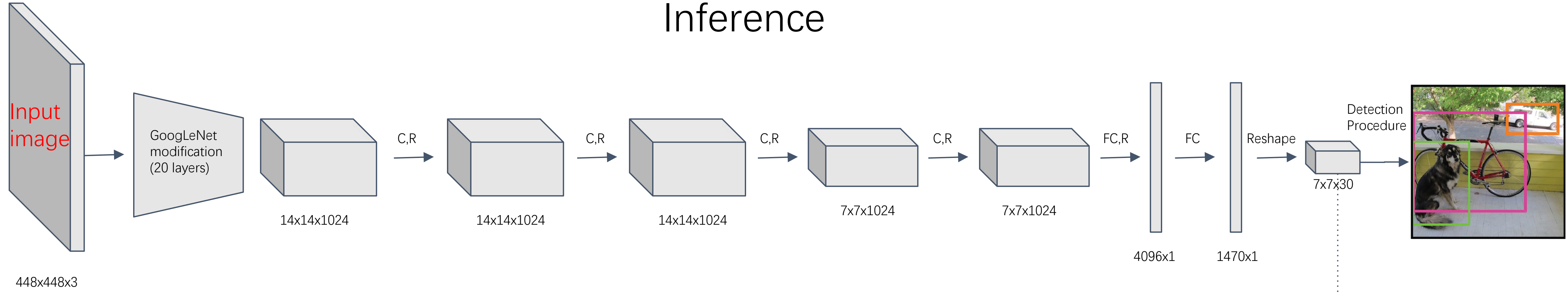


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
2 bboxes for first cell (1, 1)
bb1
bb2
20x1
20x1
7
7
30
grid cell (1, 1)
7
7

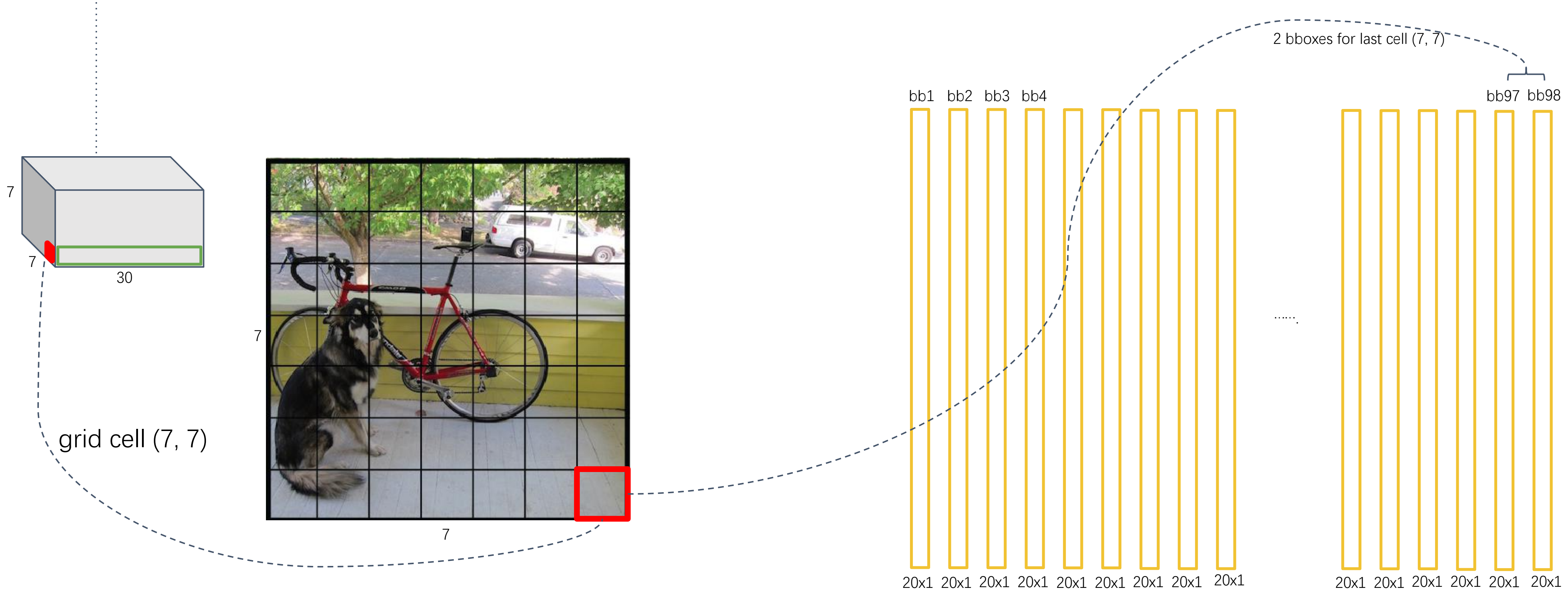
Inference


Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
2 bboxes for second cell (1, 2)
7
7
30
grid cell (1, 2)
7
7
bb1 bb2 bb3 bb4
20x1 20x1 20x1 20x1

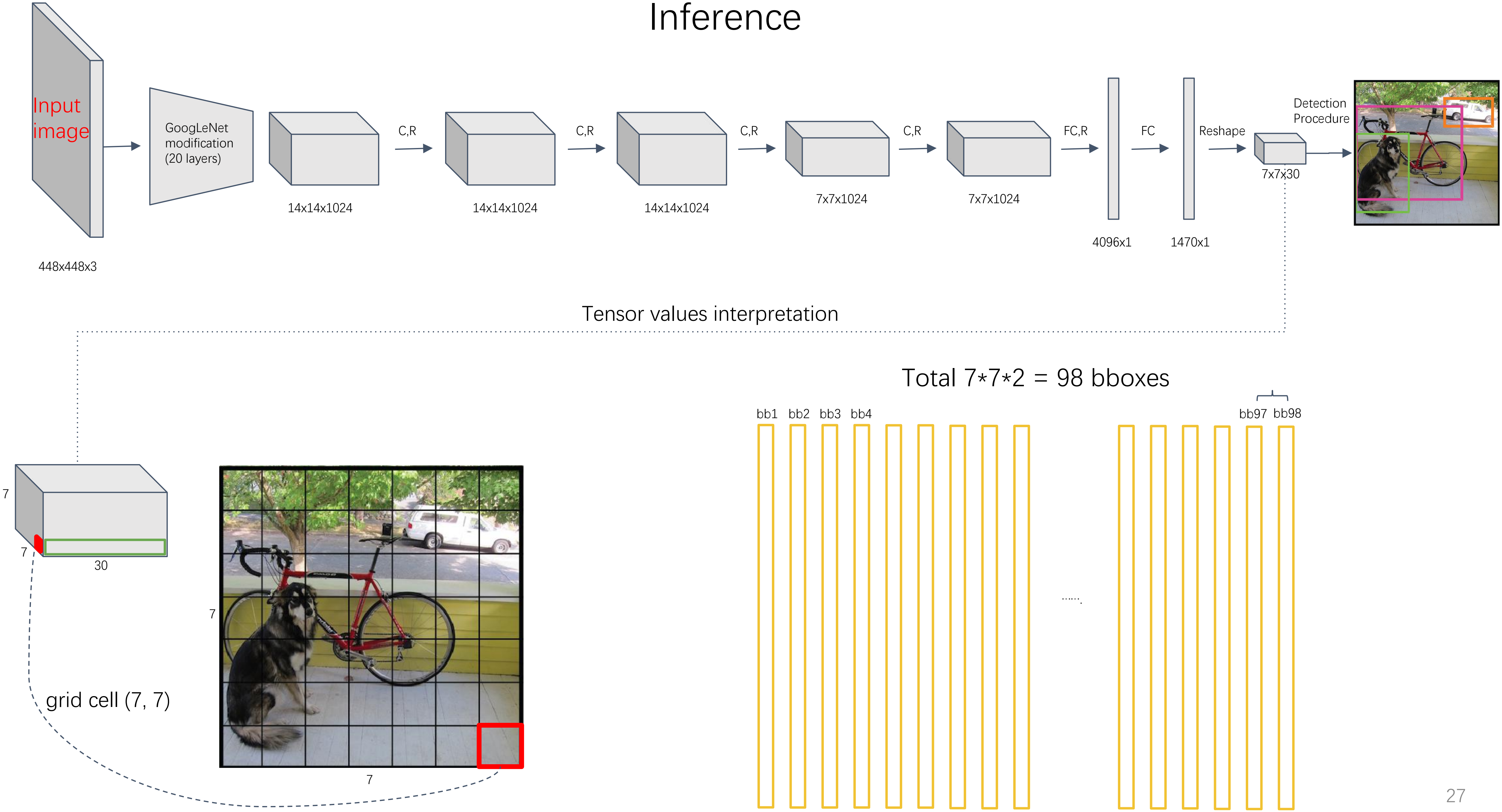
Inference

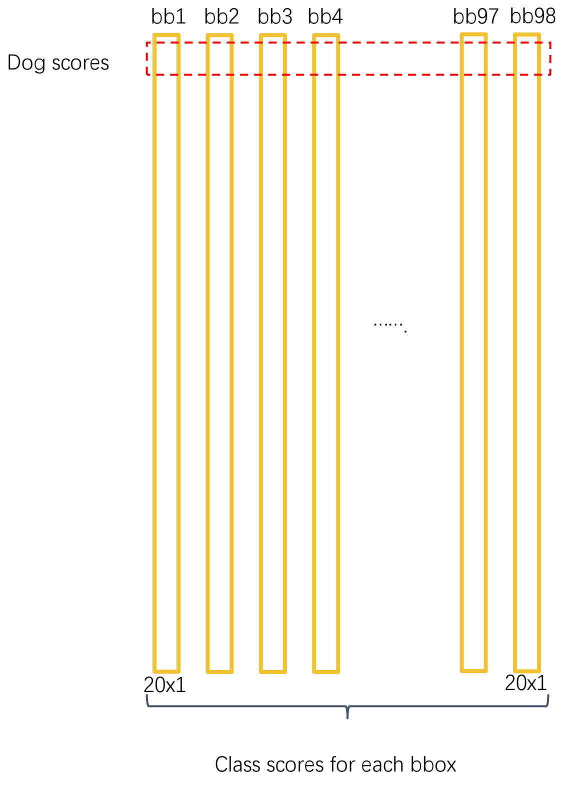


Tensor values interpretation

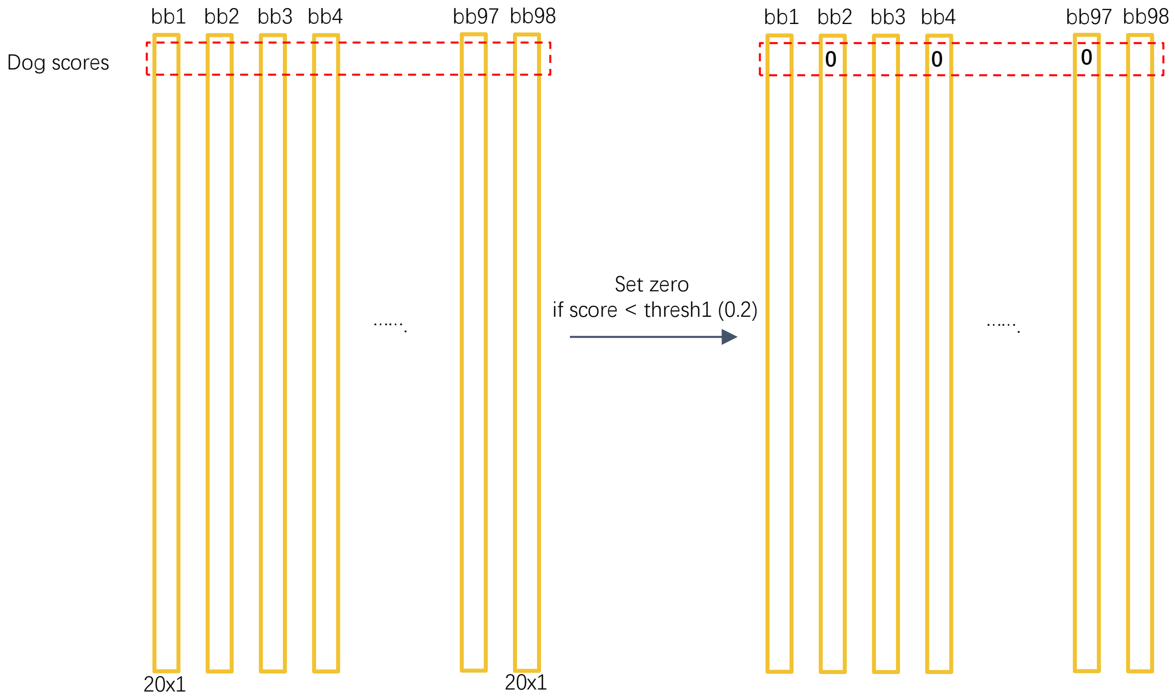


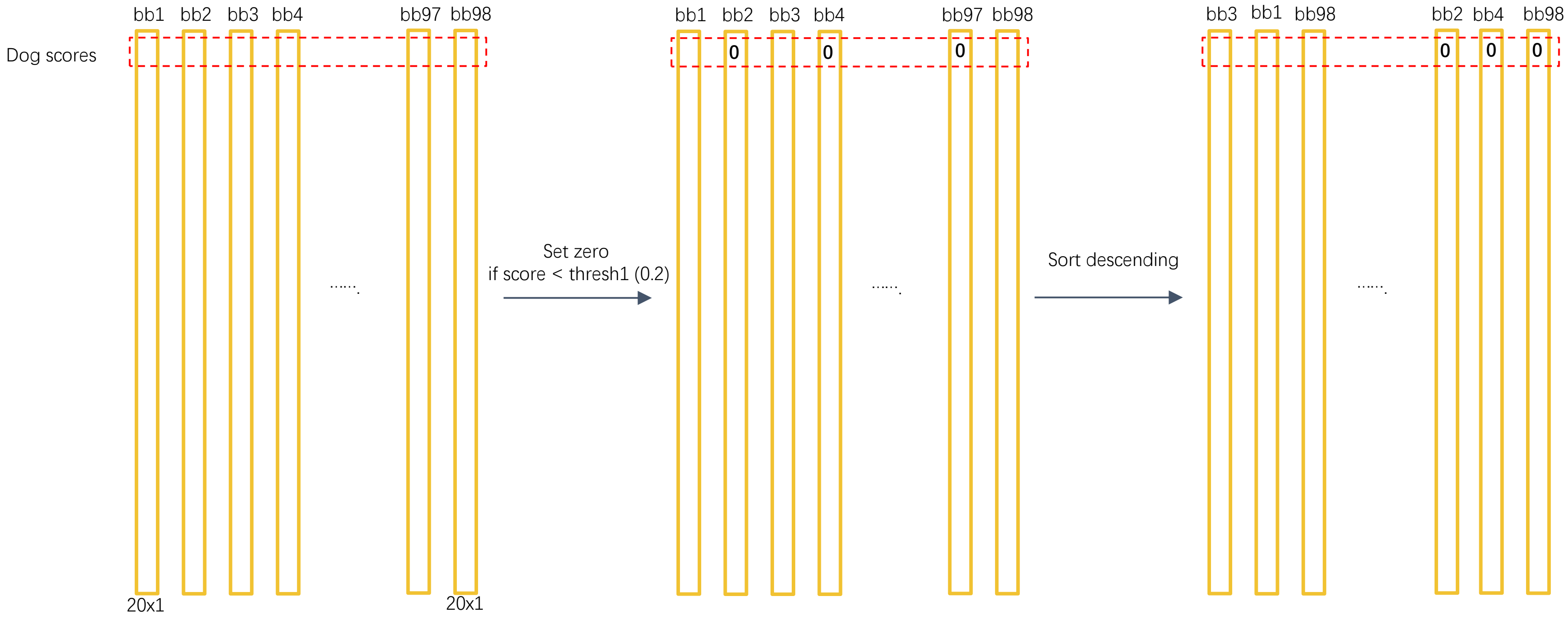
Inference

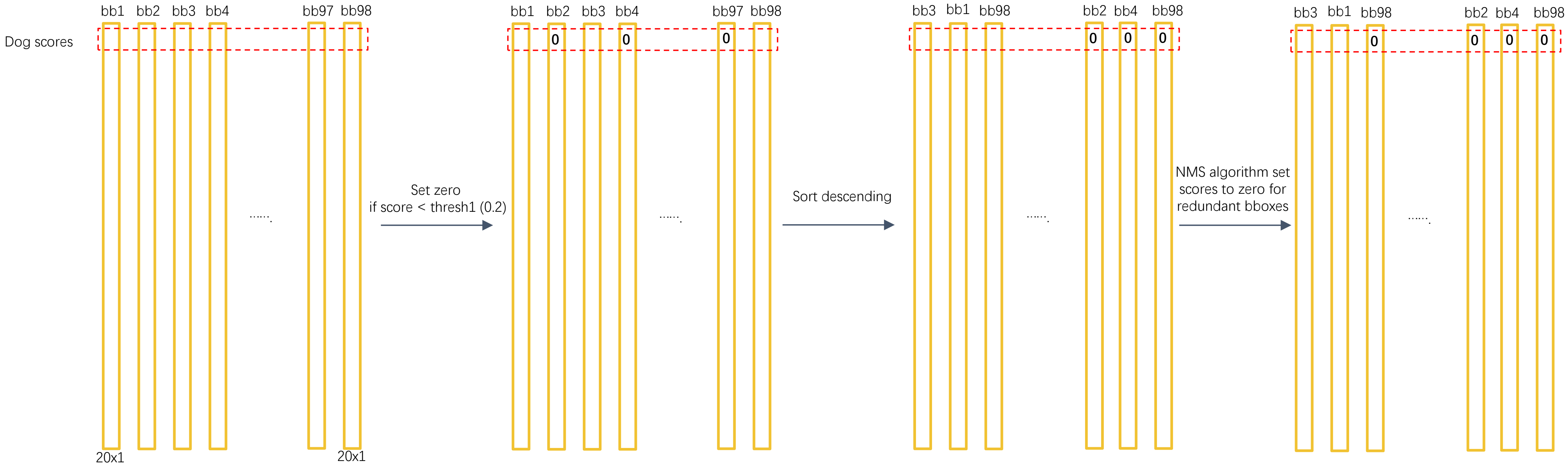

Input image
448x448x3
GoogLeNet modification (20 layers)
14x14x1024
C,R
14x14x1024
C,R
14x14x1024
C,R
7x7x1024
C,R
7x7x1024
FC,R
4096x1
FC
1470x1
Reshape
7x7x30
Detection Procedure
Tensor values interpretation
7
7
30
7
7
grid cell (7, 7)
Total 7*7*2 = 98 bboxes
bb1 bb2 bb3 bb4
bb97 bb98
……

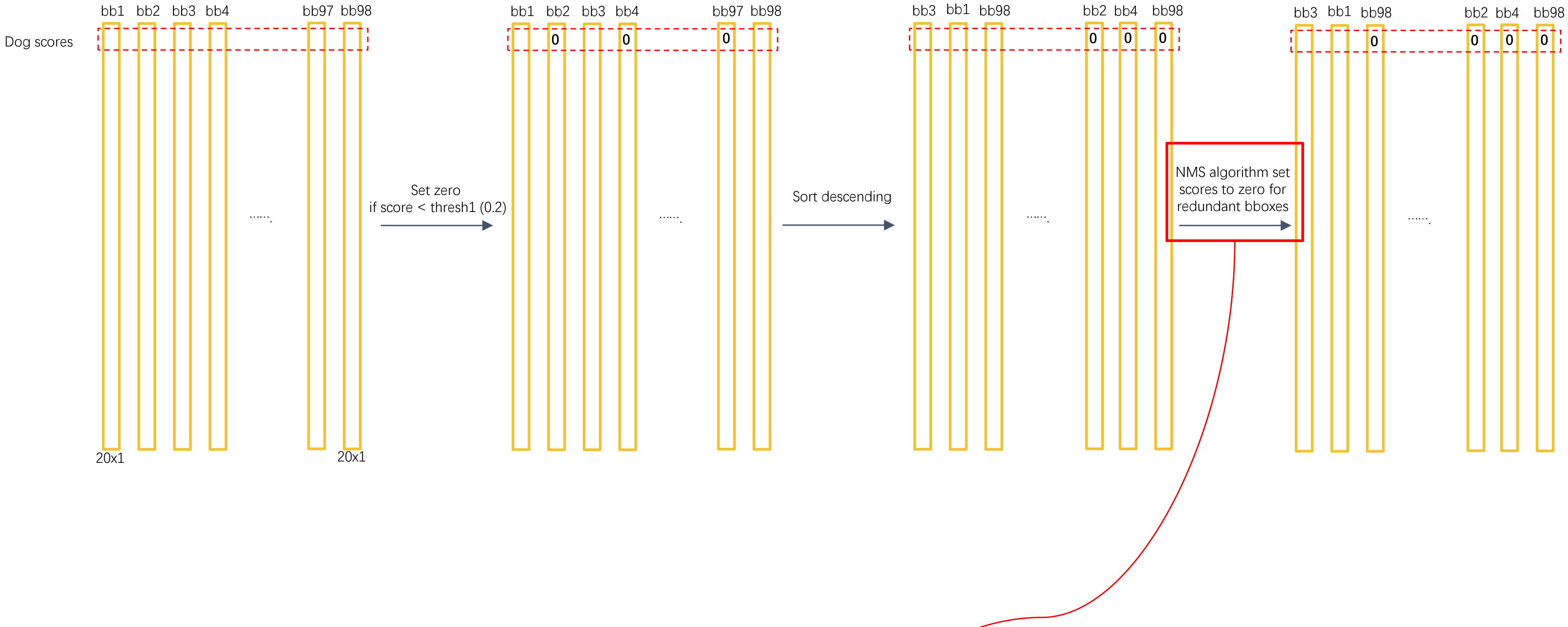


Get first class scores for each bbox


Dog scores
bb1
bb2
bb3
bb4
bb97
bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1
bb2
bb3
bb4
bb97
bb98
0
0
0


Dog scores
bb1 bb2 bb3 bb4 bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
0
0
0
Sort descending
bb3 bb1 bb98 bb2 bb4 bb98
0
0
0


Dog scores
bb1 bb2 bb3 bb4 bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
0 0 0
Sort descending
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0 0
……


Dog scores
bb1 bb2 bb3 bb4 bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
0 0 0
Sort descending
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0
NMS algorithm set scores to zero for redundant bboxes
bb3 bb1 bb98 bb2 bb4 bb98
0 0 0 0

How it works

Non-Maximum Suppression: intuition

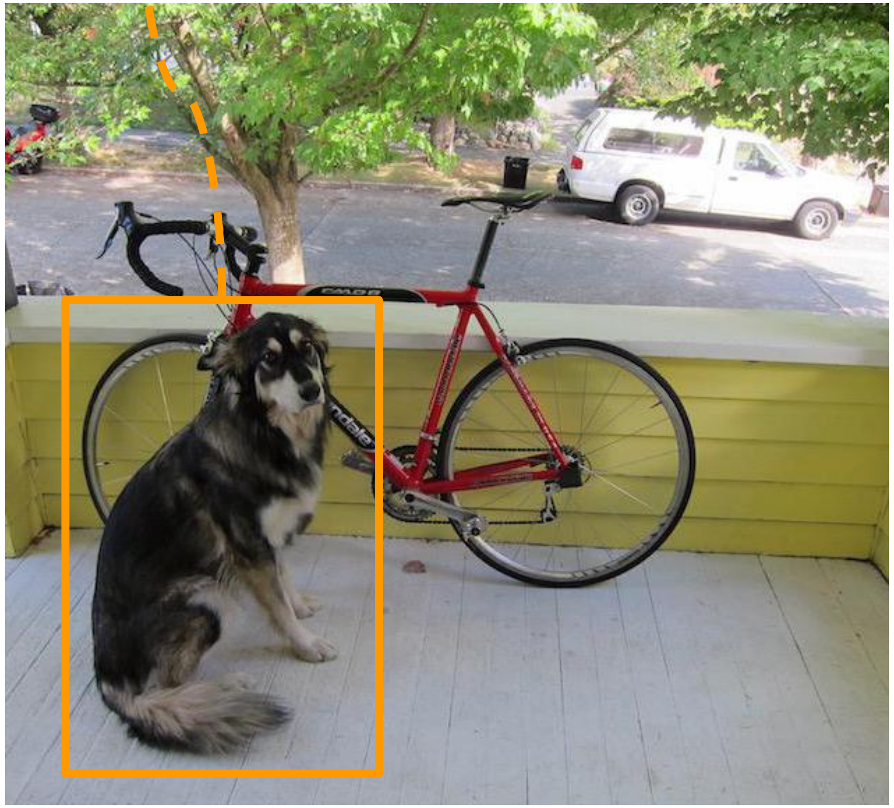
class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1							0	0	0	0	1x98

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

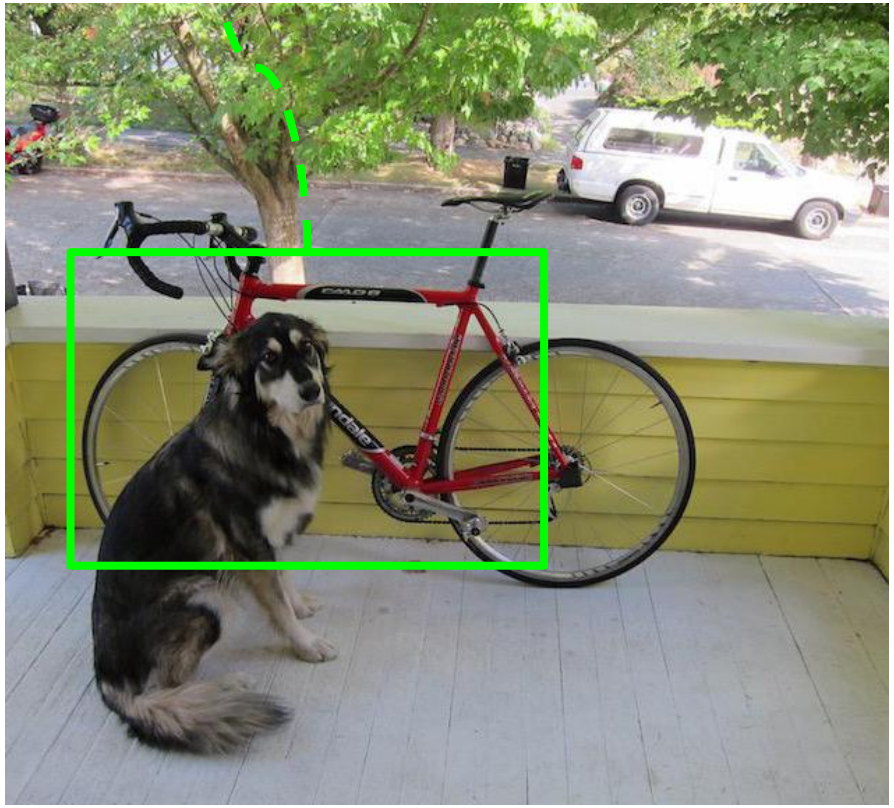
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1							0	0	0	0	1x98



Non-Maximum Suppression: intuition

class (dog) scores for each bbox

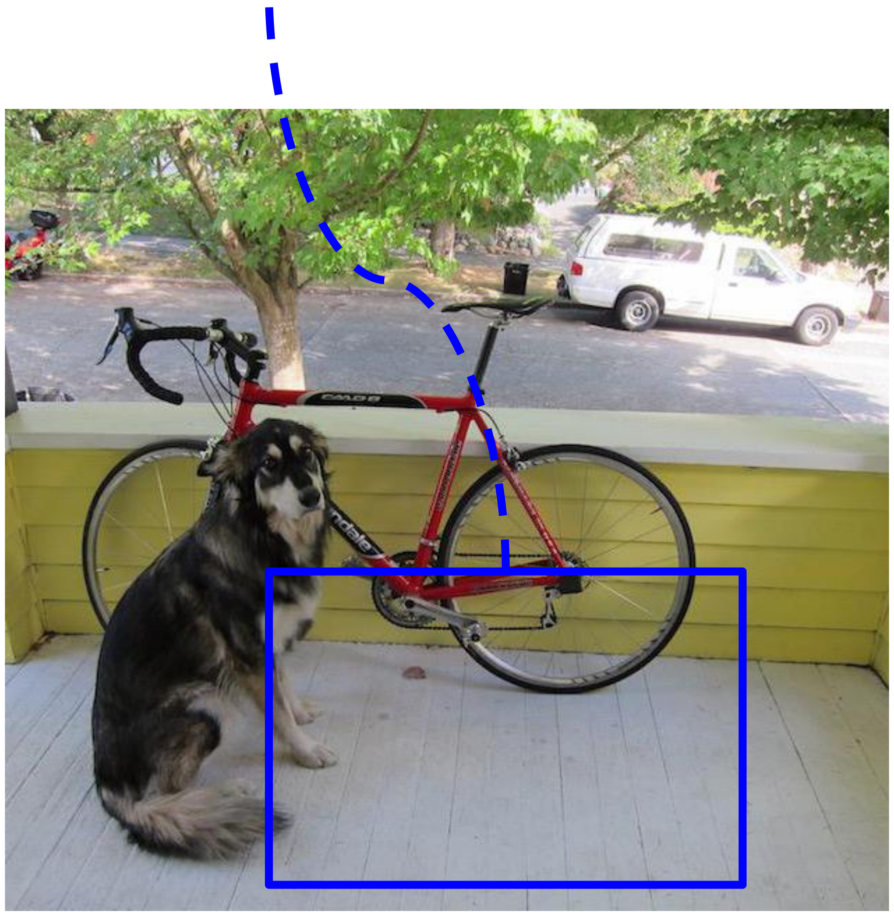
	bb47	bb20	bb15	bb7								bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1								0	0	0	0	1x98



Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1							0	0	0	0	1x98



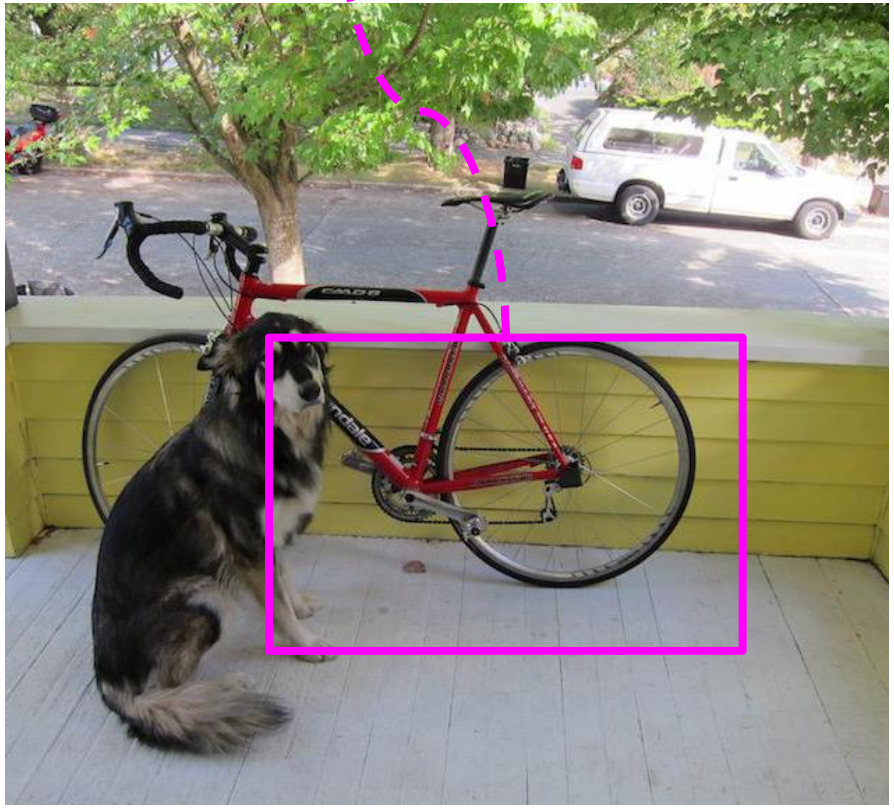
Non-Maximum Suppression: intuition

class (dog) scores for each bbox

class: dog

bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98
0.5	0.3	0.2	0.1							0	0	0	0

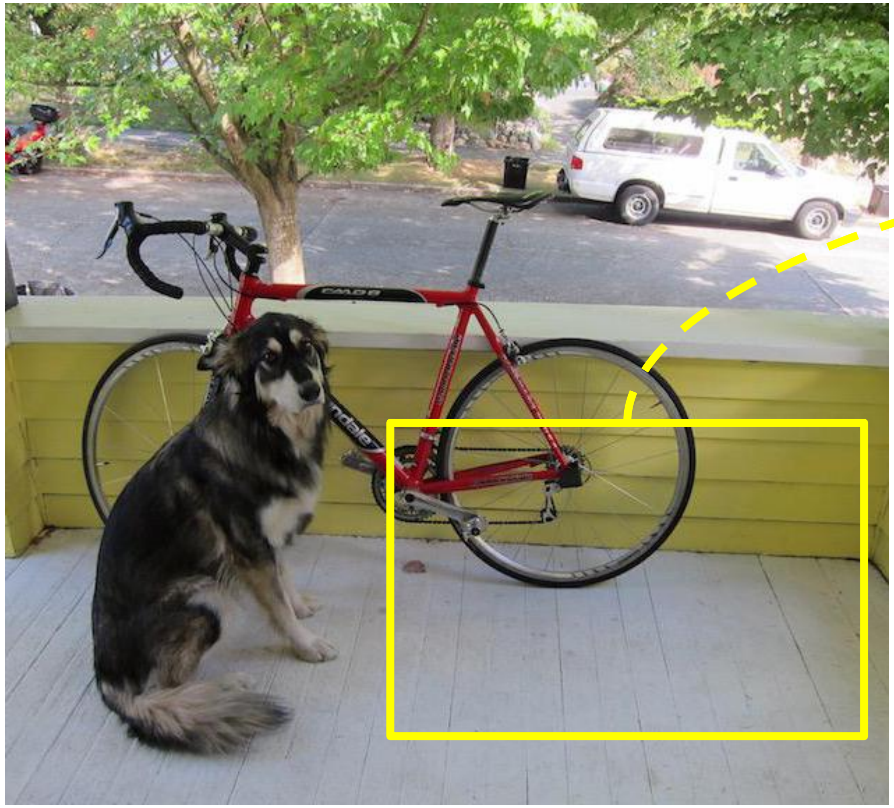
1x98



Non-Maximum Suppression: intuition

class (dog) scores for each bbox

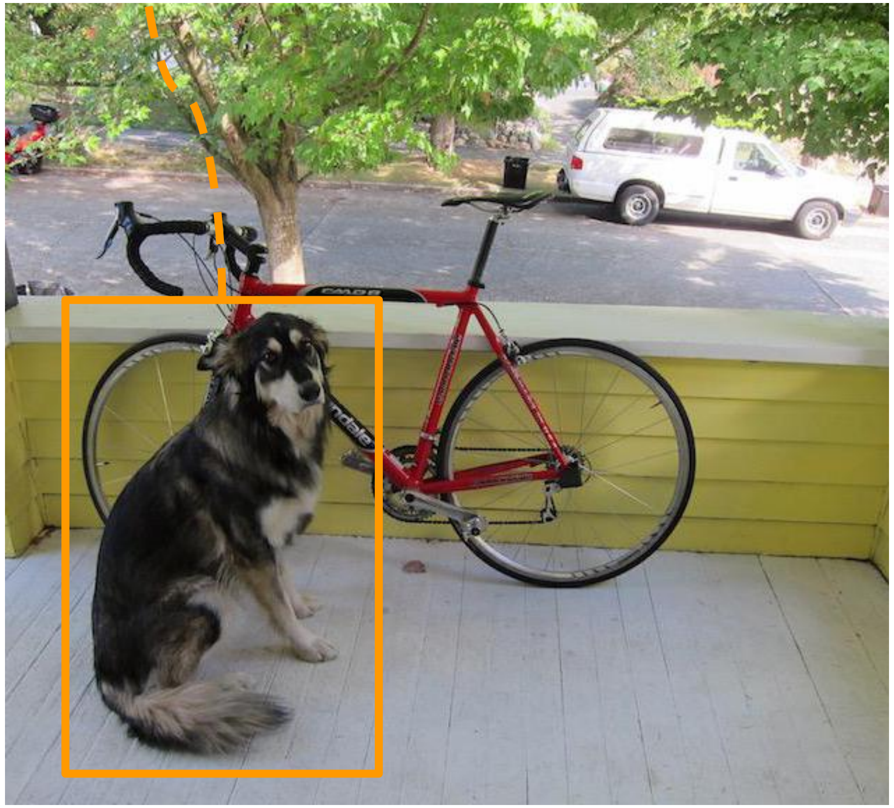
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1							0	0	0	0	1x98



Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1							0	0	0	0	1x98

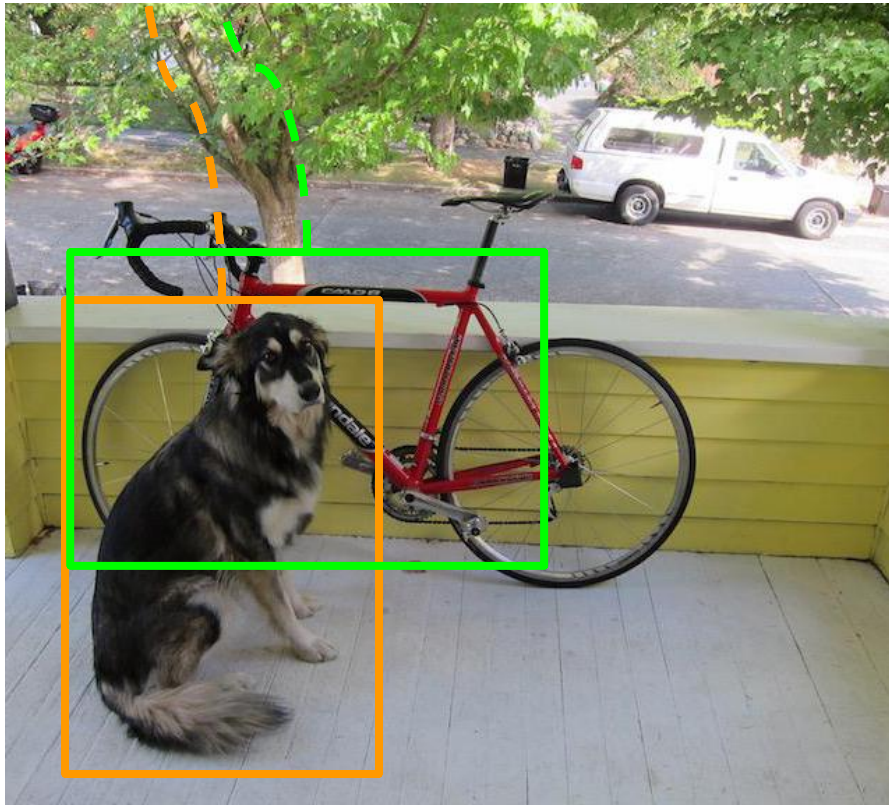


Get bbox with max score.
Let's denote it
"bbox_max"

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7		bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1		0	0	0	0	1x98

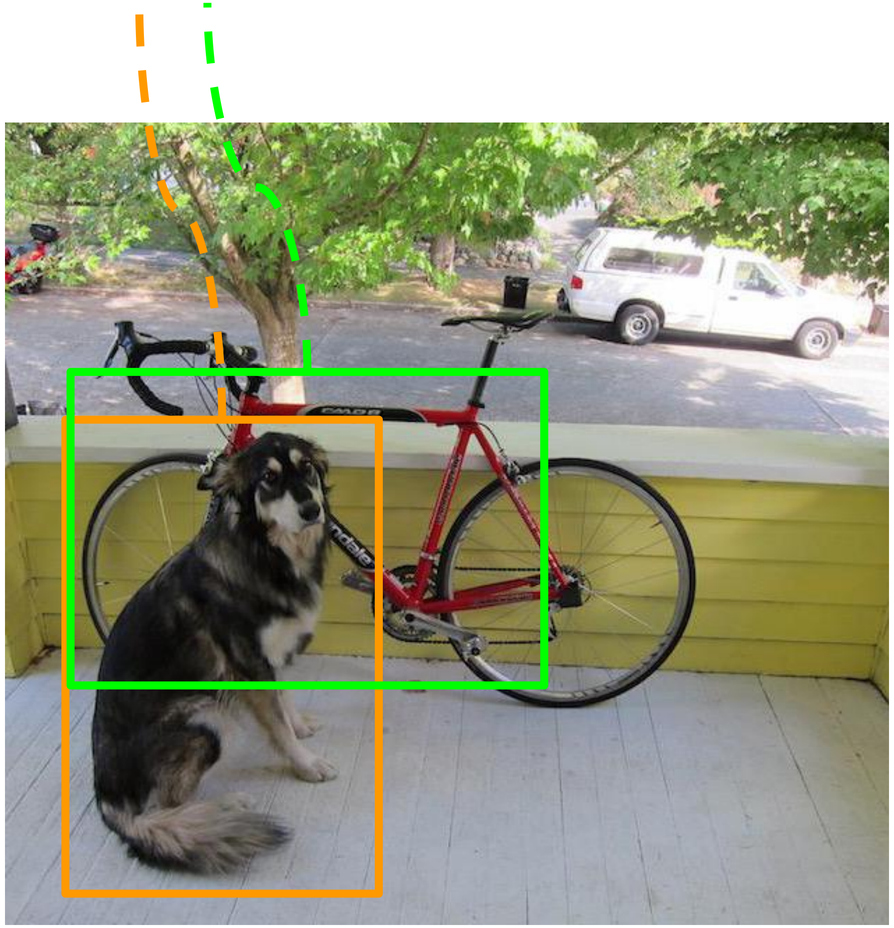


Compare "bbox_max" with others less score (non-zero!) bboxes. Let's denote it "bbox_cur"

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0.3	0.2	0.1							0	0	0	0	1x98

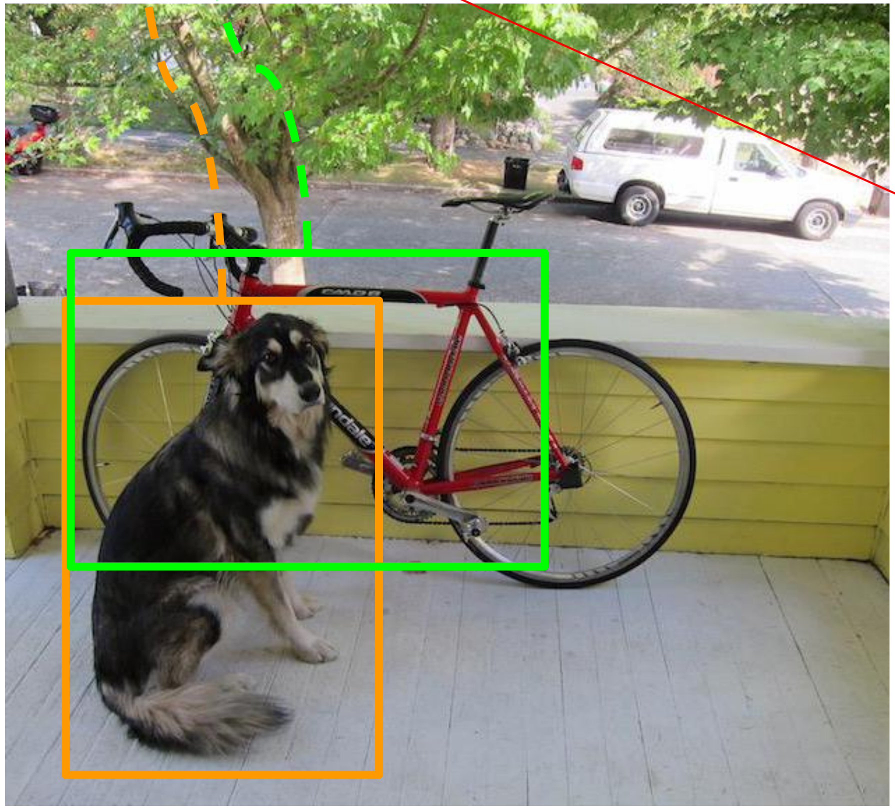


If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98



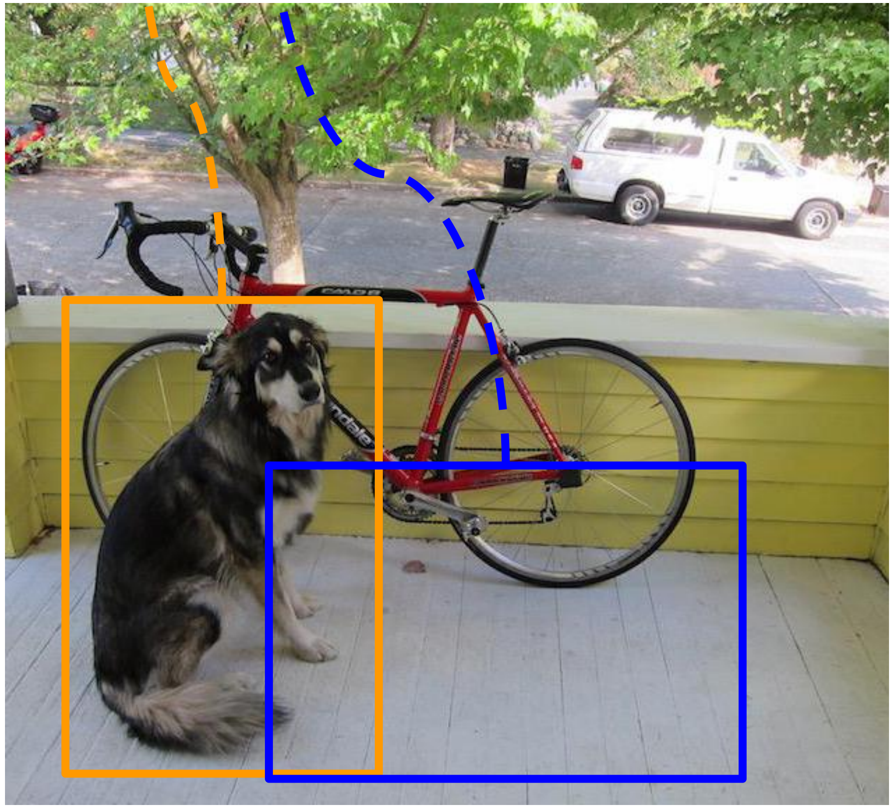
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: set to 0.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98

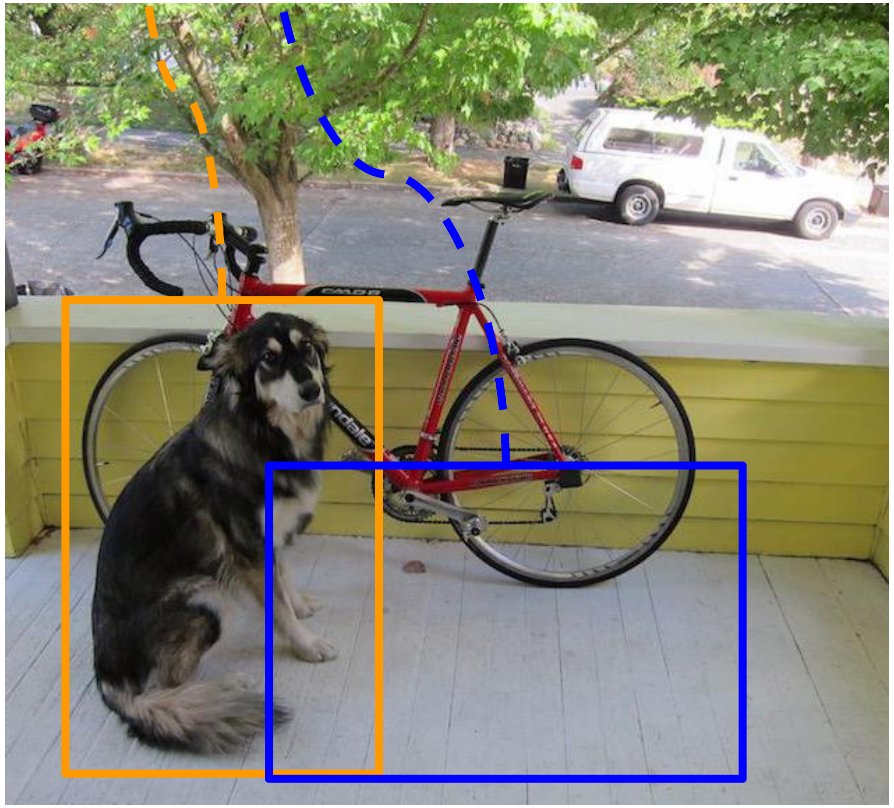


Go to next bbox_cur.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	**0**	0.2	0.1							0	0	0	0	1x98



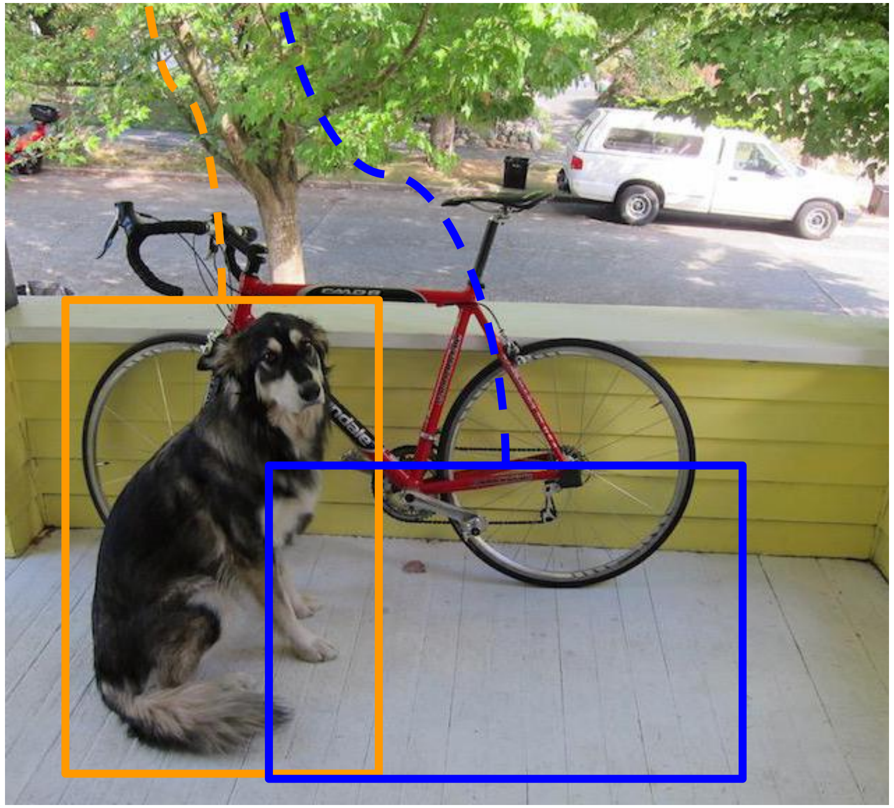
Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98



Go to next bbox_cur.

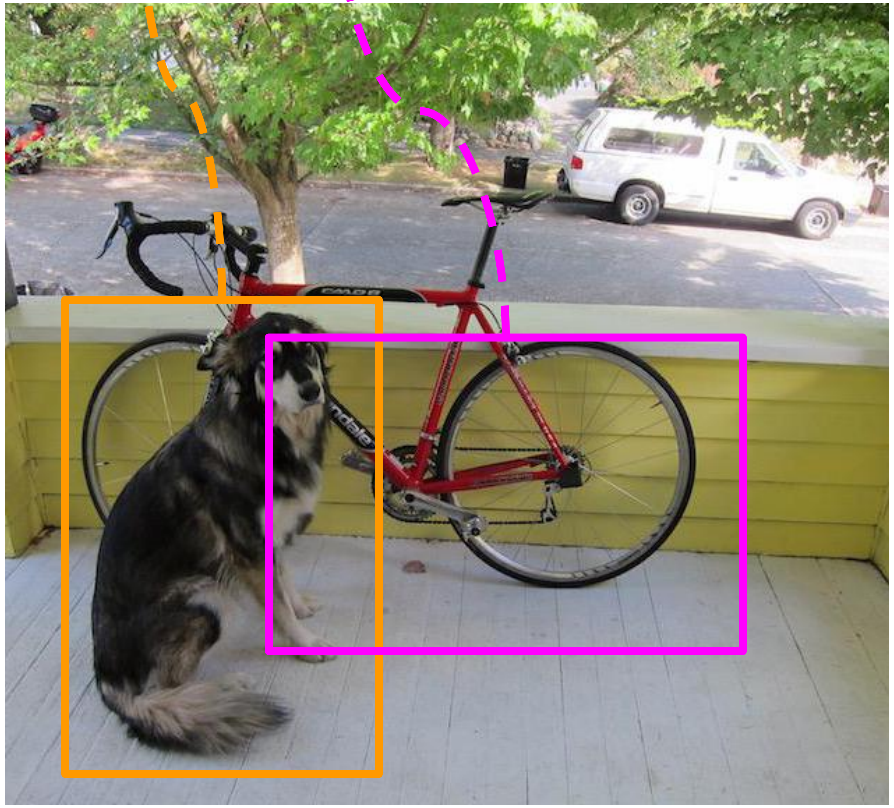
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: continue.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98

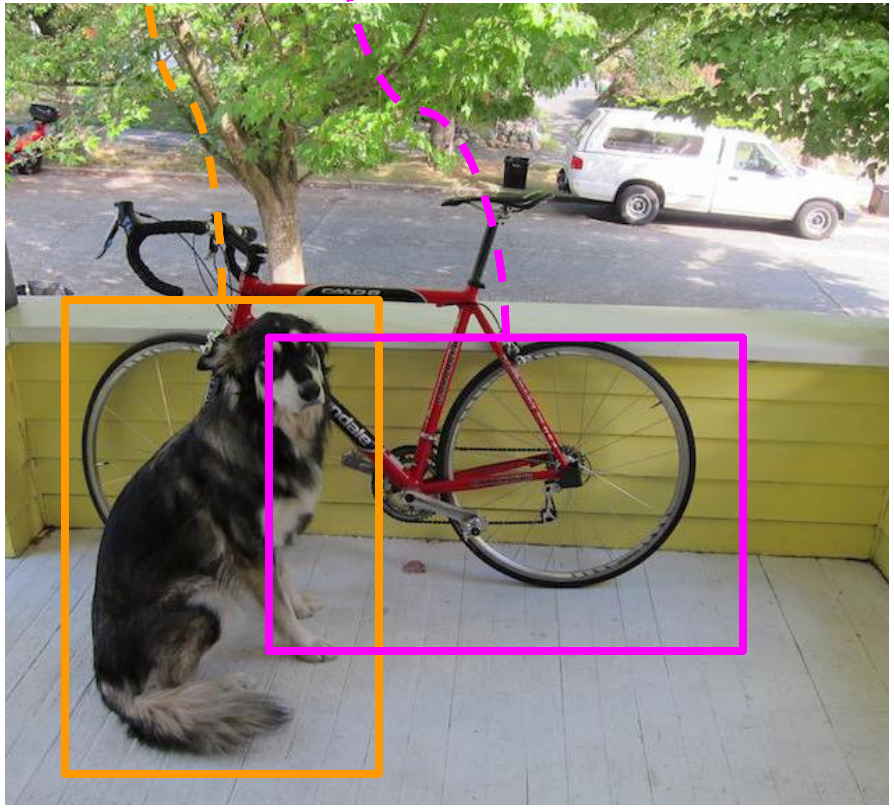


Go to next bbox_cur.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	**0**	0.2	0.1							0	0	0	0	1x98



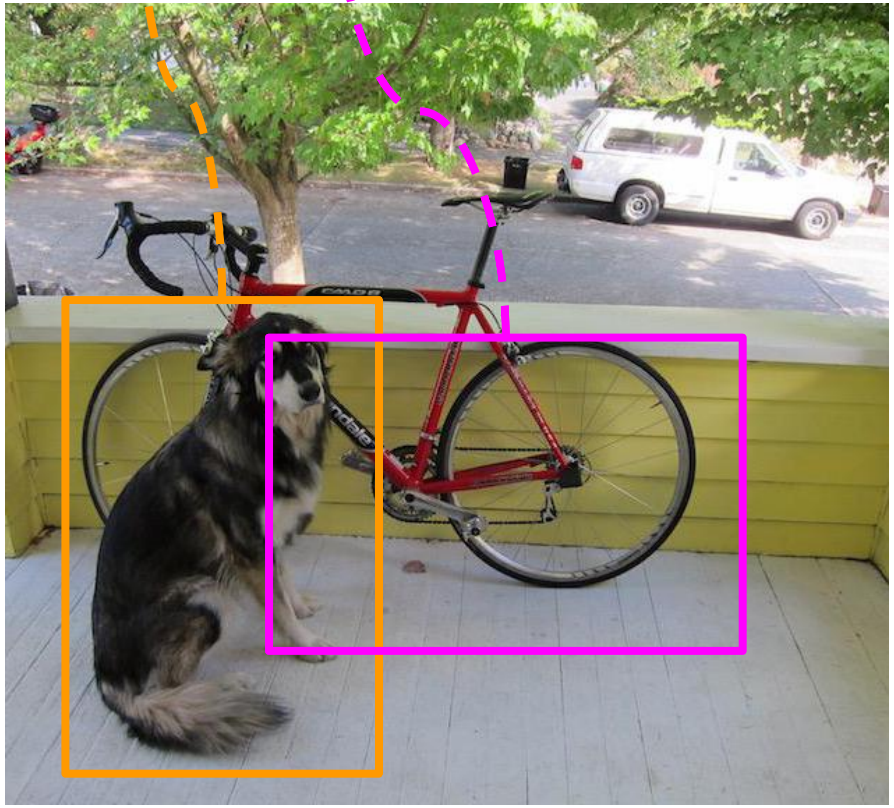
Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	**0**	0.2	0.1							0	0	0	0	1x98



Go to next bbox_cur.

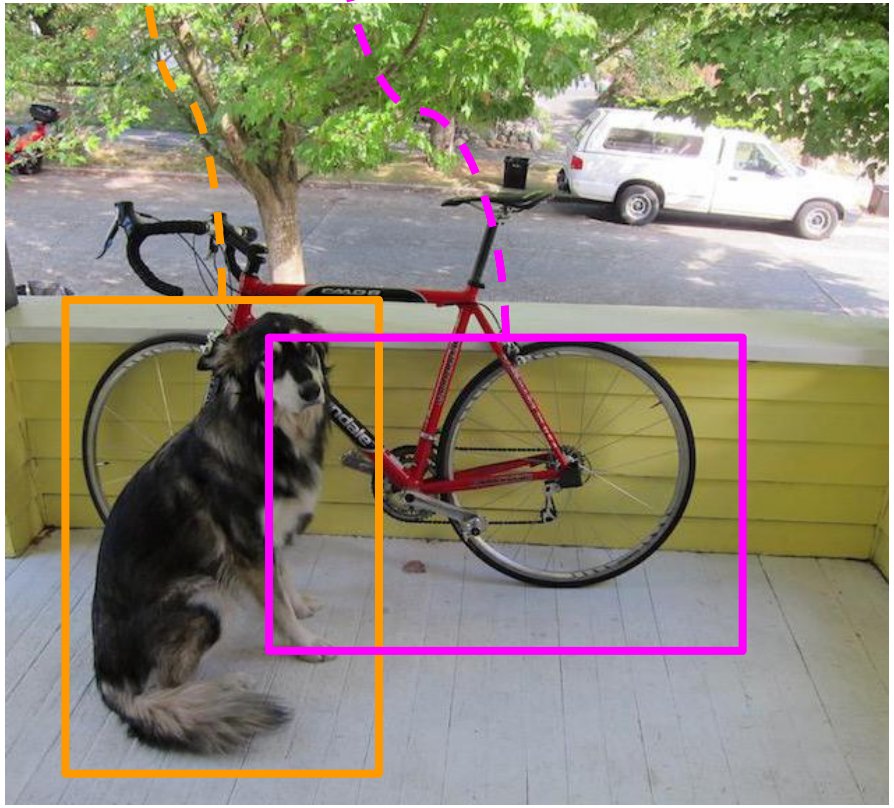
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

In this case: continue.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98



Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

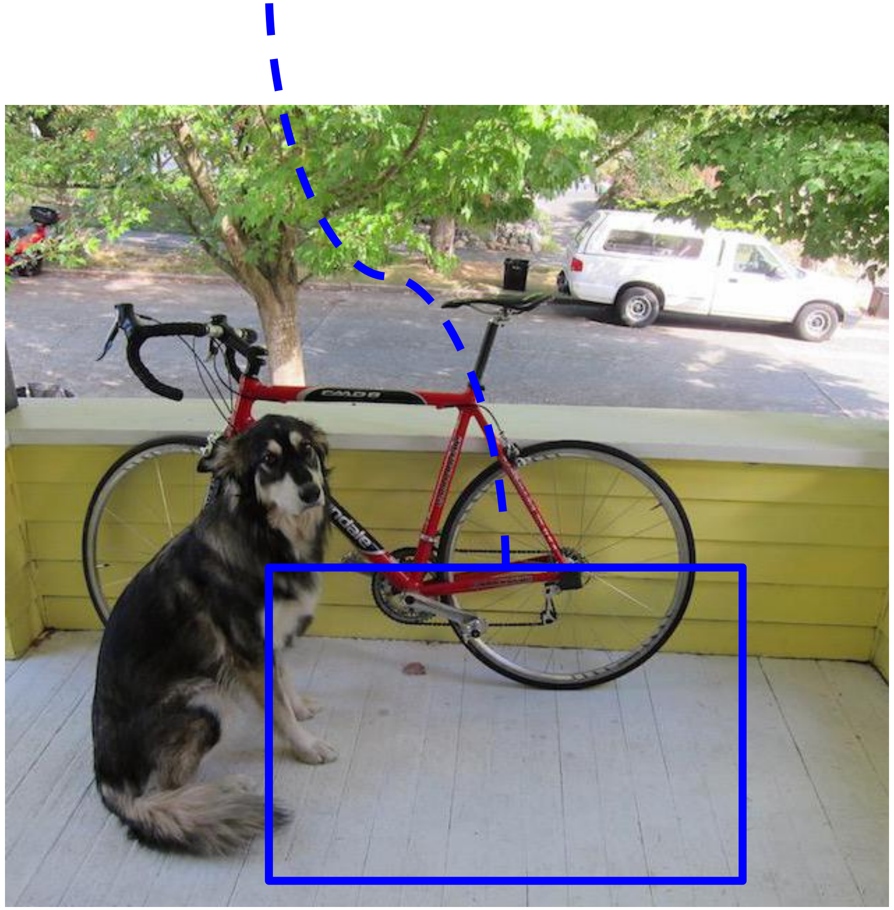
In this case: continue.

Do this procedure for other “bbox_cur”.
After that …

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98

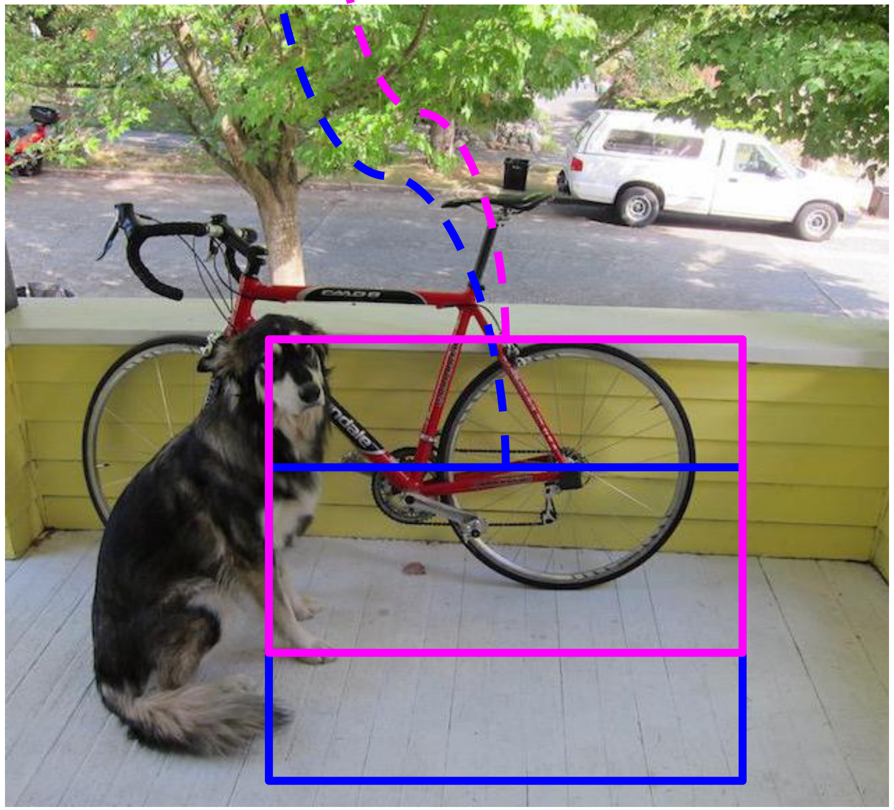


Go to next bbox with big score. Let’s denote it “bbox_max”

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98

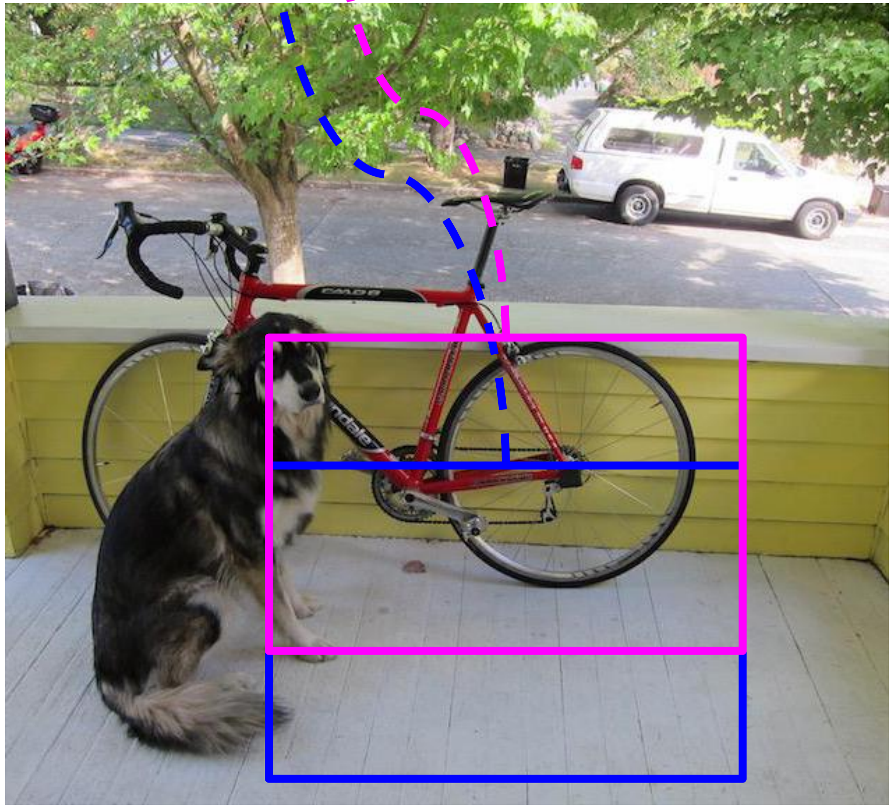


Go to next bbox_cur.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0.1							0	0	0	0	1x98



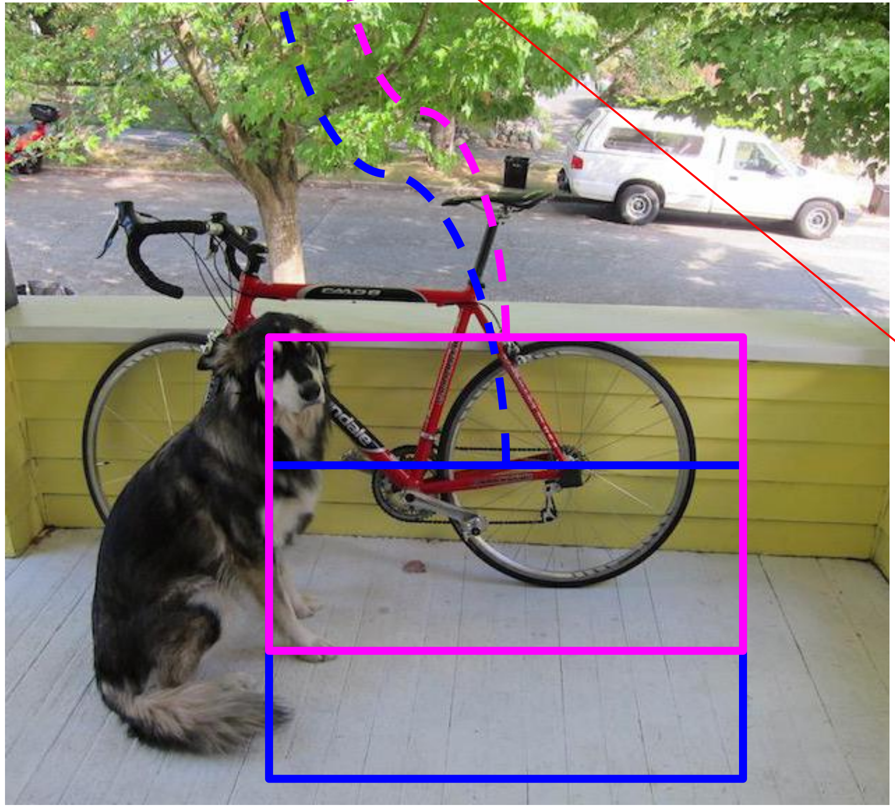
Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

Non-Maximum Suppression: intuition

class (dog) scores for each bbox

	bb47	bb20	bb15	bb7		bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	**0**		0	0	0	0	1x98



Go to next bbox_cur.

If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.

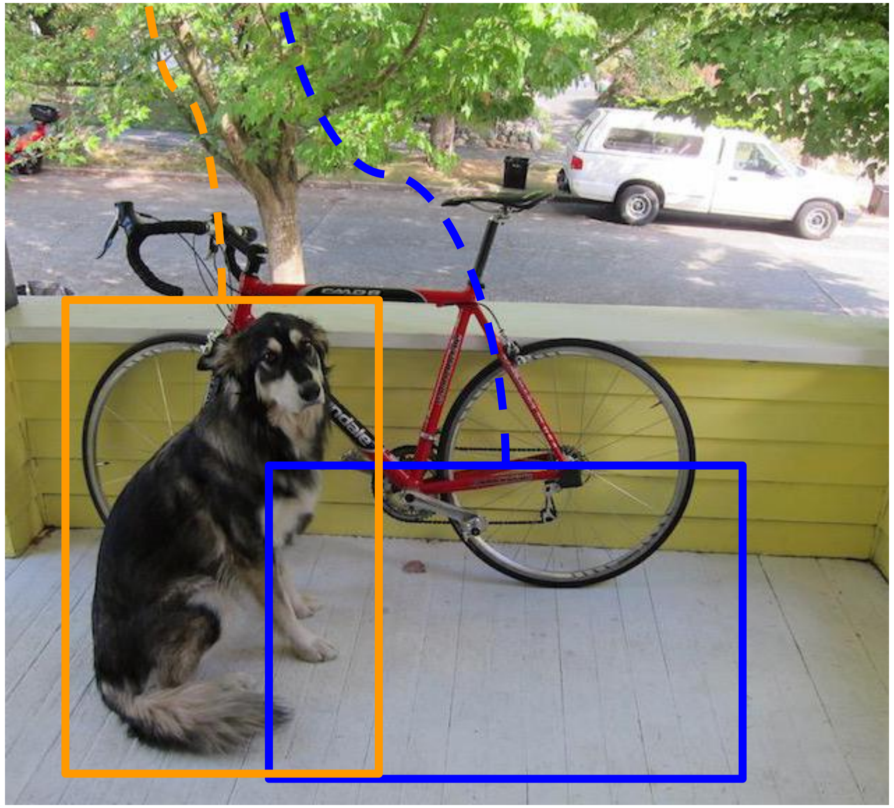
In this case: set to 0.

Do this procedure for other “bbox_max” and for other corresponding “bbox_cur”.

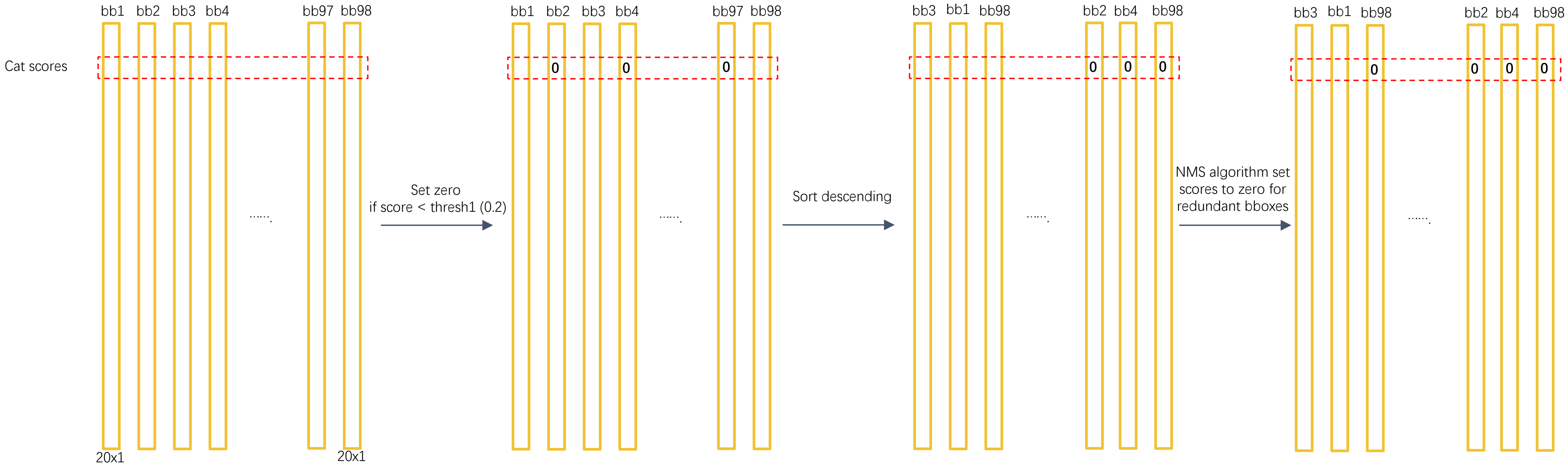
Non-Maximum Suppression: intuition

class (dog) scores for each bbox

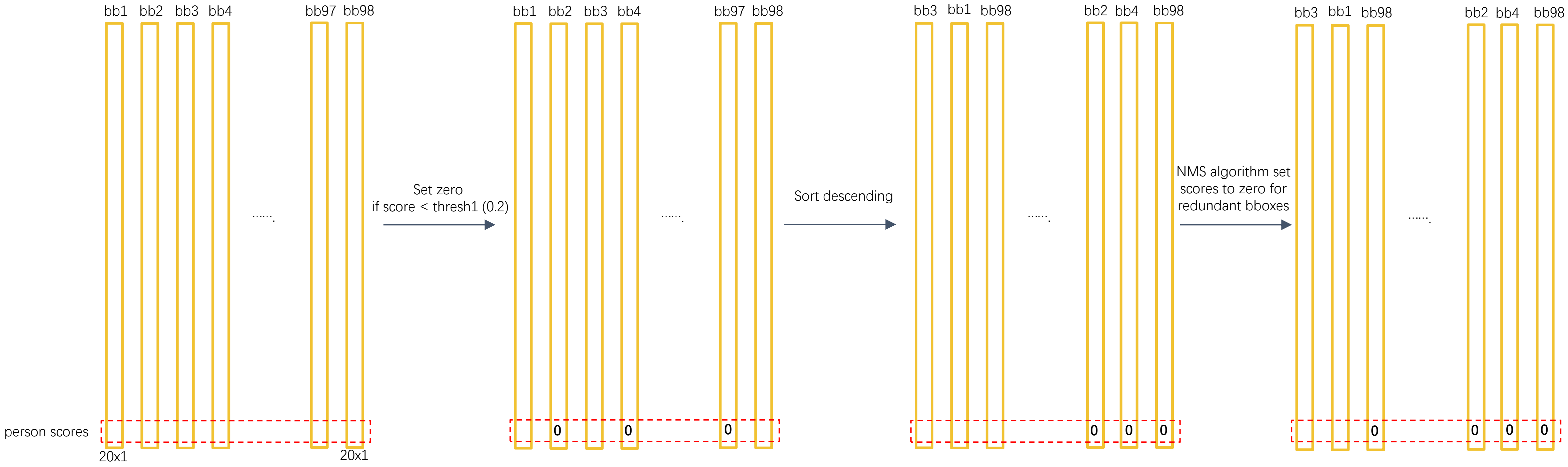
	bb47	bb20	bb15	bb7							bb1	bb4	bb8	bb98	
class: dog	0.5	0	0.2	0							0	0	0	0	1x98



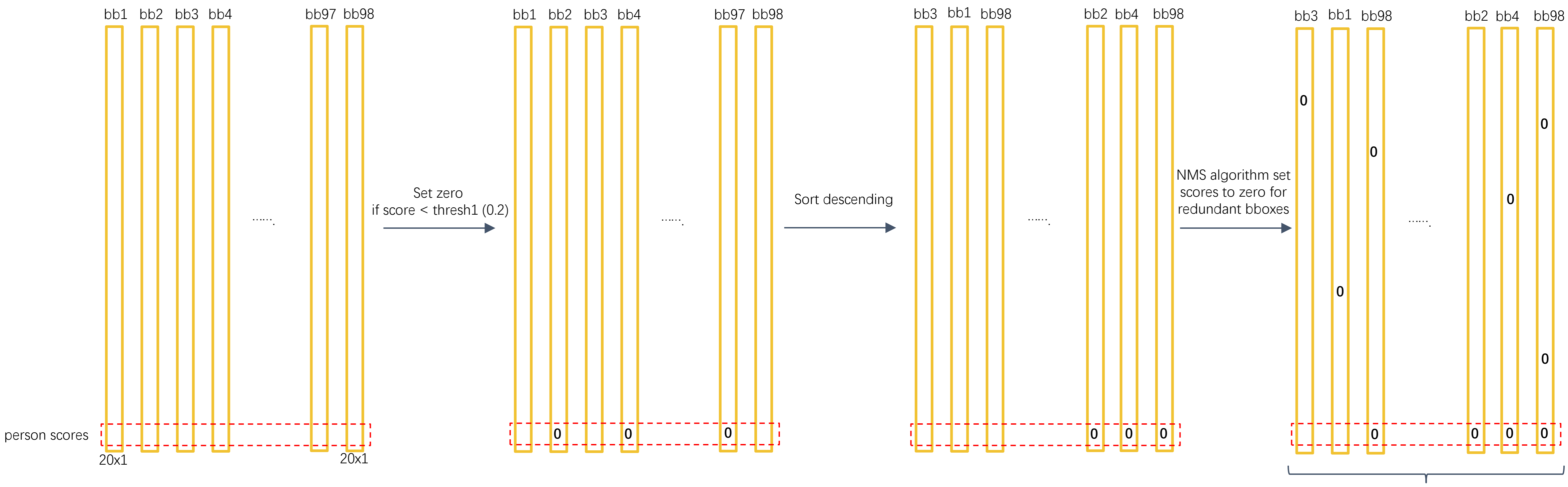
After comparison almost all pairs of bboxes the only two bboxes left with non-zero class score value.



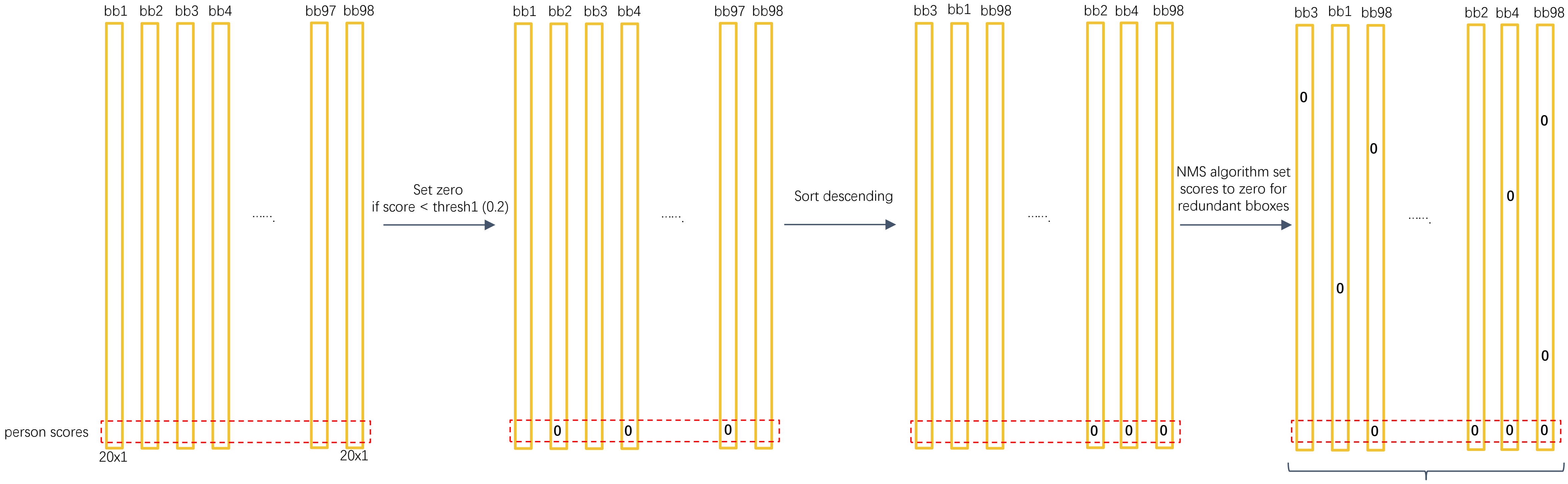
Do this procedure for next class



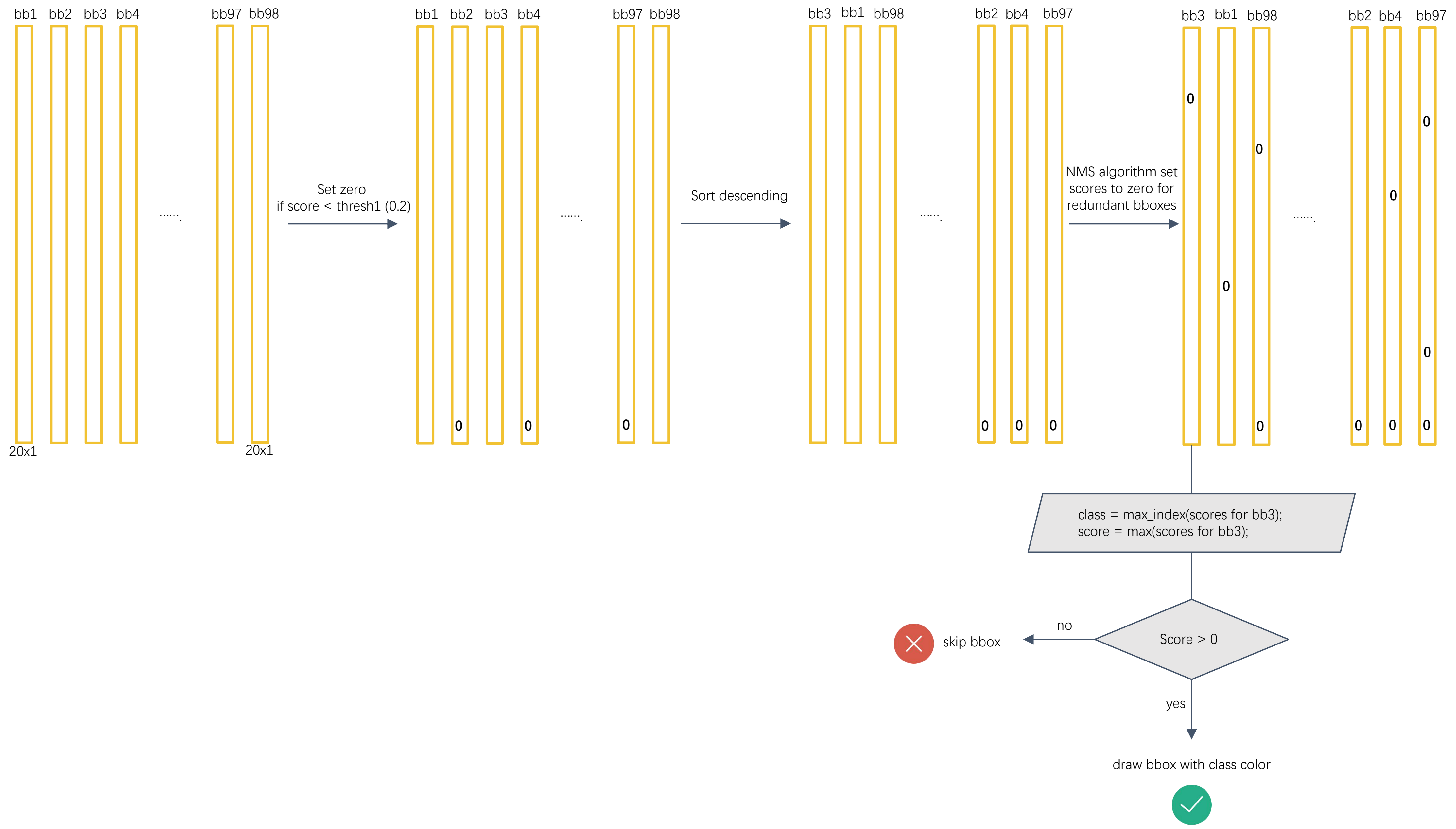
Do this procedure for all classes



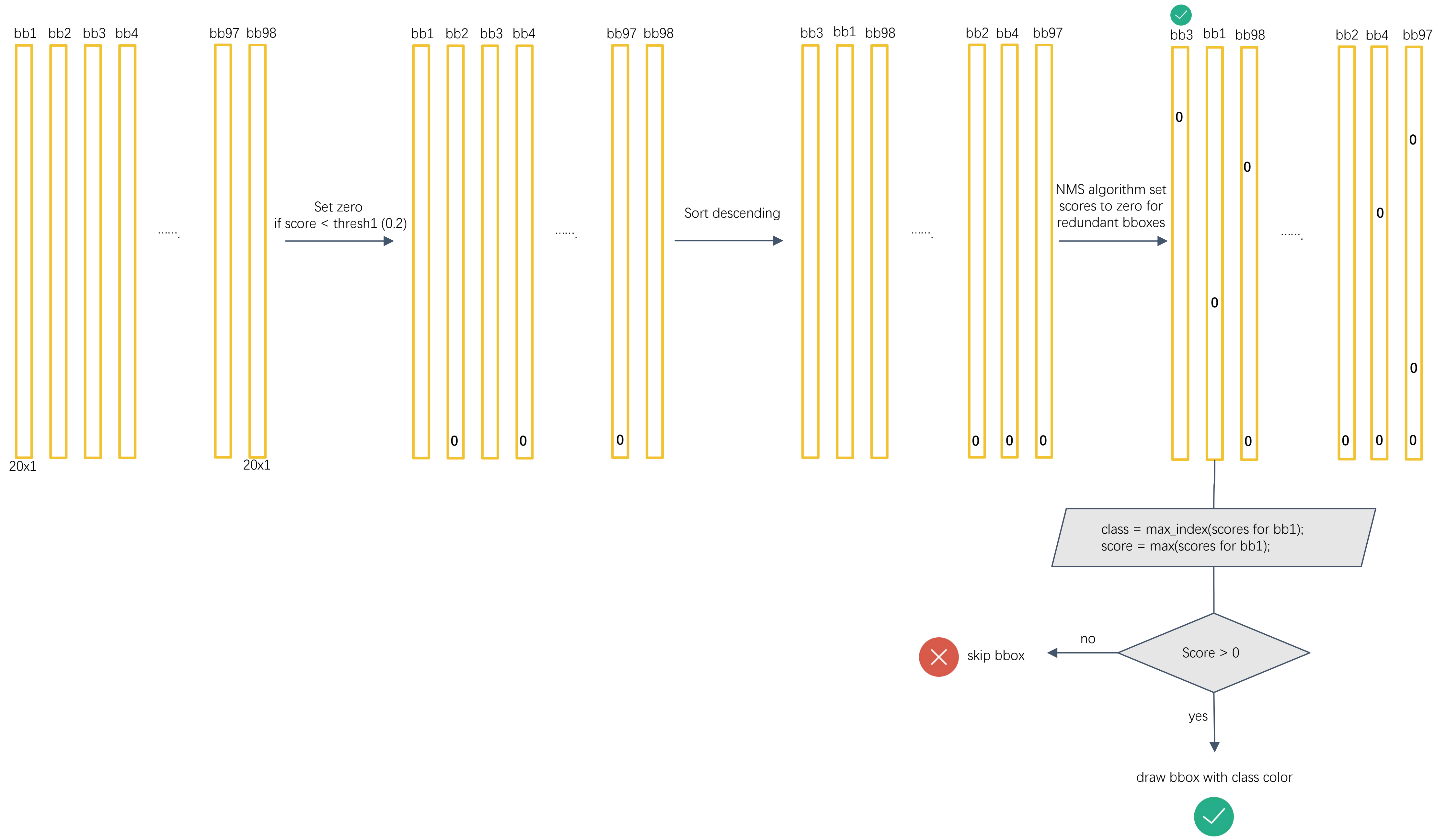
After this procedure -
a lot of zeros

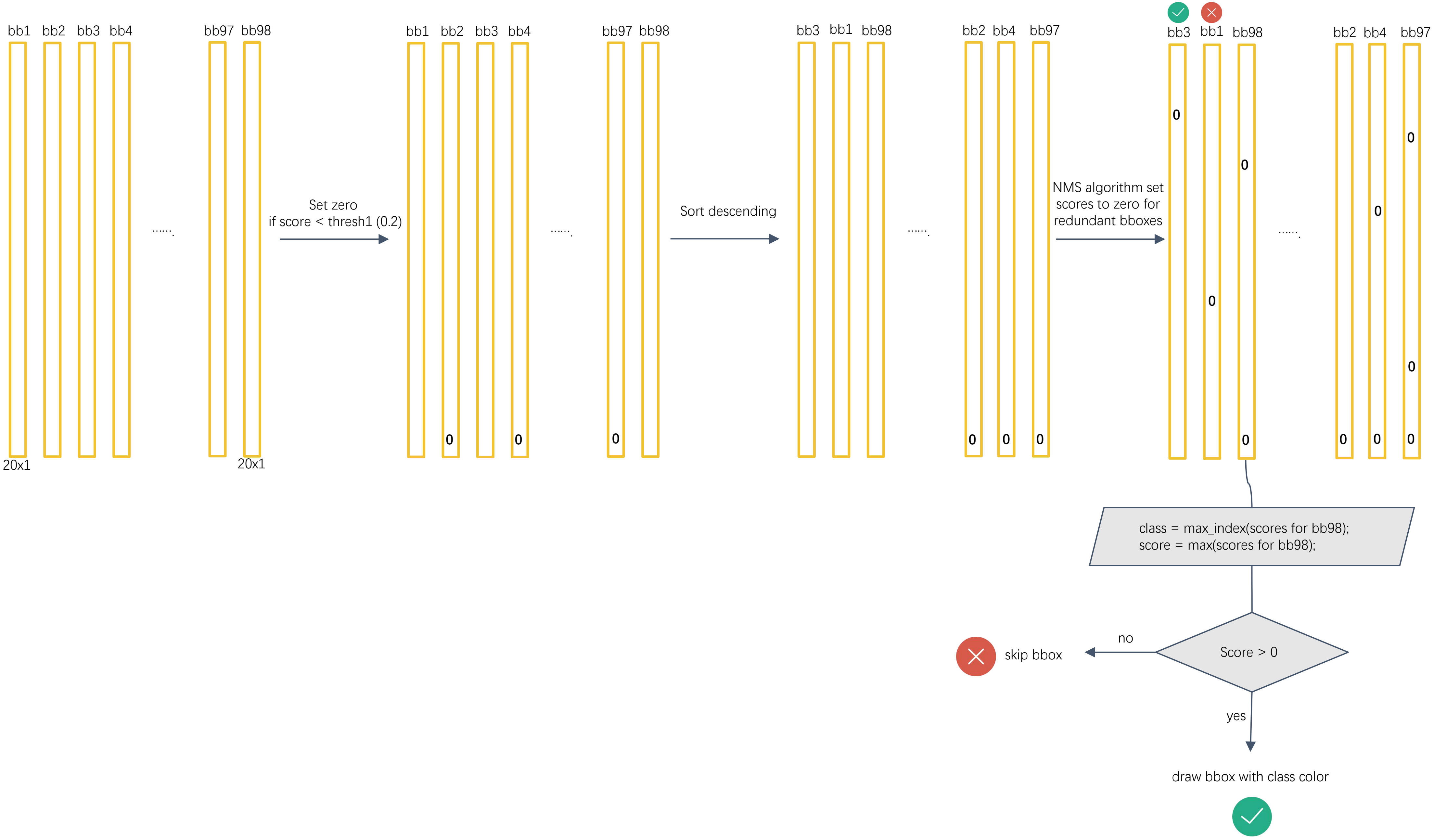


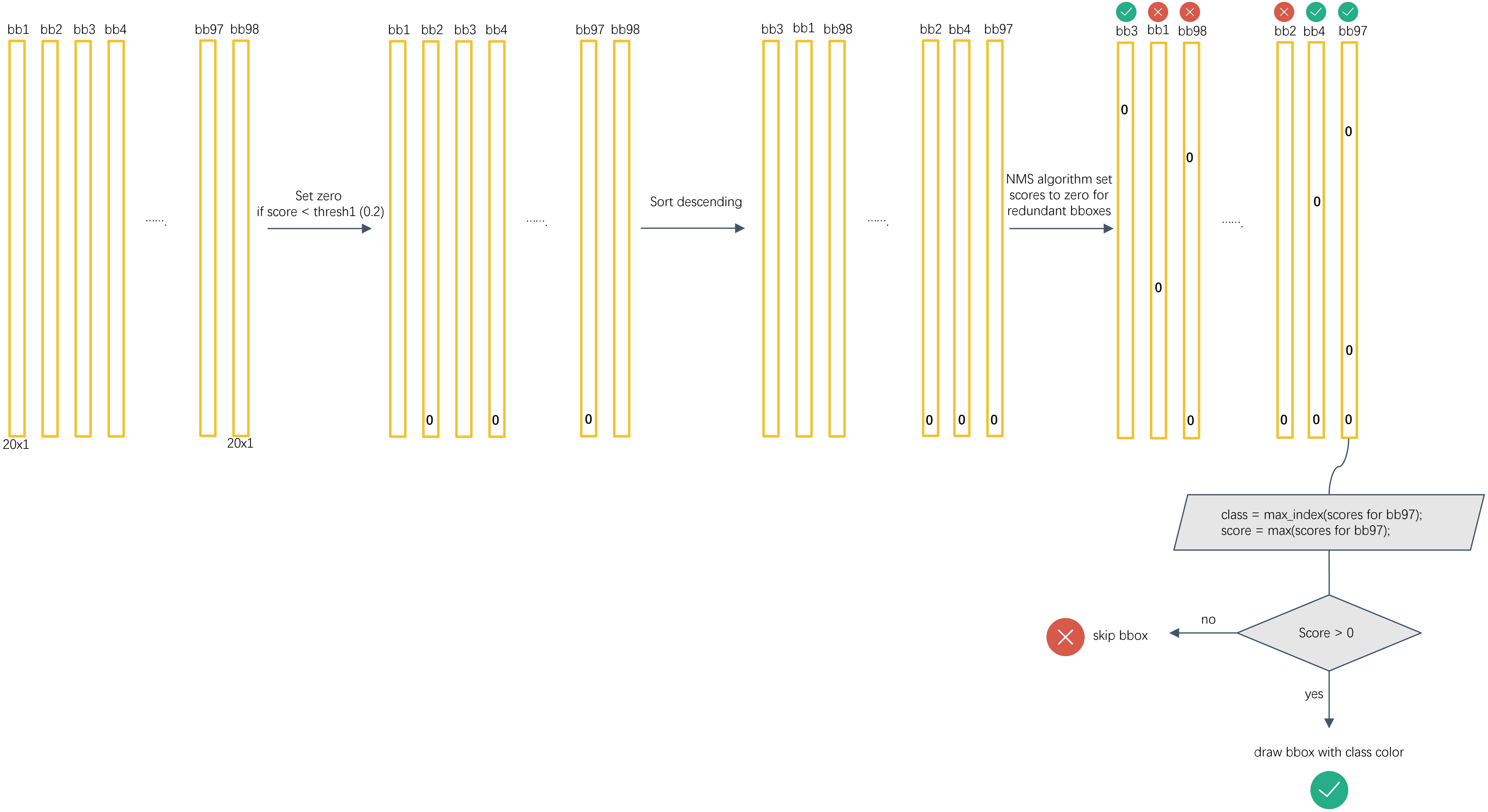
Select bboxes to draw by class score values

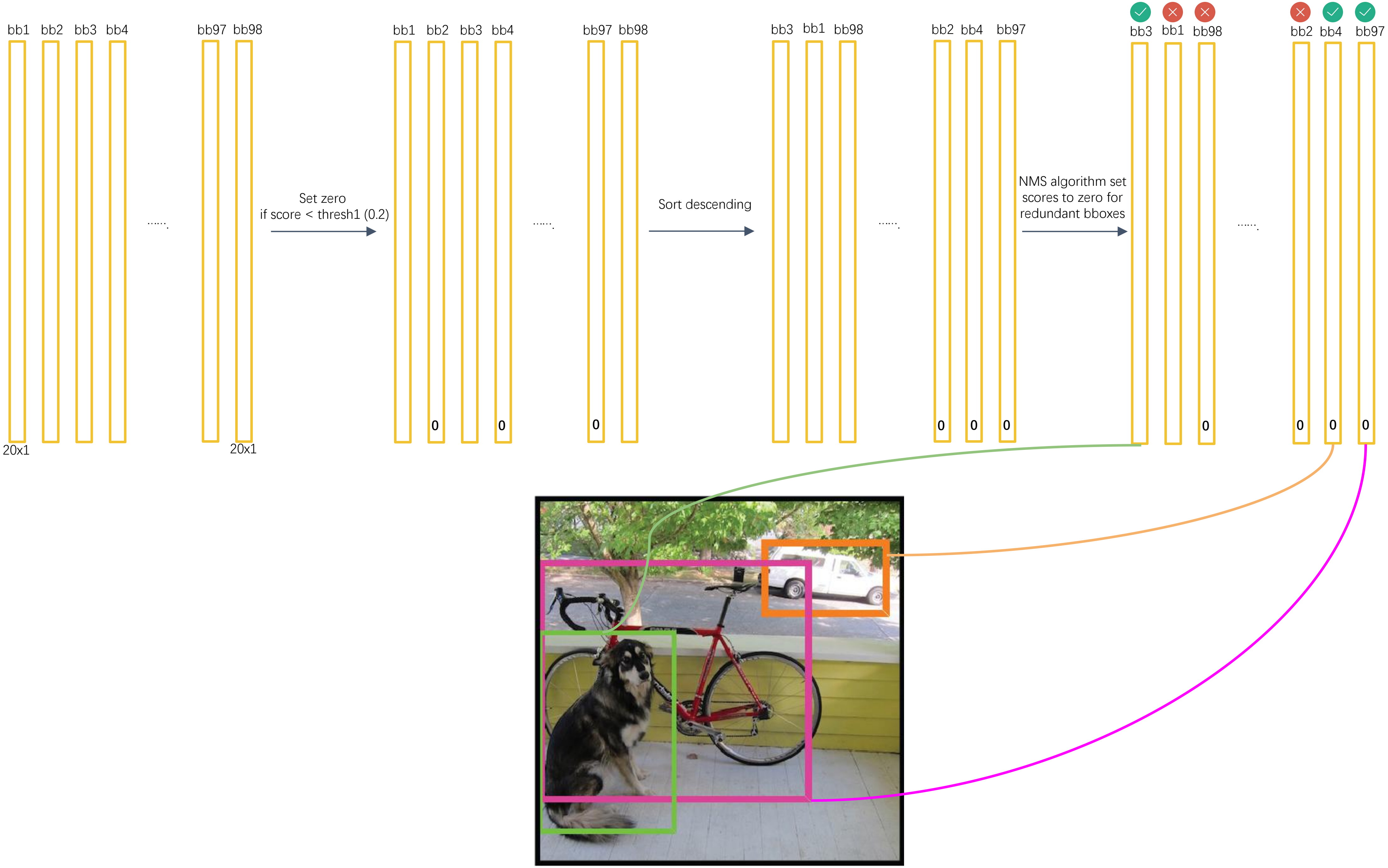


bb1 bb2 bb3 bb4
bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4
bb97 bb98
Sort descending
bb3 bb1 bb98
bb2 bb4 bb97
NMS algorithm set scores to zero for redundant bboxes
bb3 bb1 bb98
bb2 bb4 bb97
class = max_index(scores for bb3);
score = max(scores for bb3);
Score > 0
no
skip bbox
yes
draw bbox with class color


bb1 bb2 bb3 bb4
bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4
bb97 bb98
Sort descending
bb3 bb1 bb98
bb2 bb4 bb97
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98
bb2 bb4 bb97
class = max_index(scores for bb1);
score = max(scores for bb1);
Score > 0
no
skip bbox
yes
draw bbox with class color


bb1 bb2 bb3 bb4
bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
Sort descending
NMS algorithm set scores to zero for redundant bboxes
class = max_index(scores for bb98);
score = max(scores for bb98);
Score > 0
no
skip bbox
yes
draw bbox with class color


bb1 bb2 bb3 bb4 bb97 bb98
20x1
20x1
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4 bb97 bb98
Sort descending
bb3 bb1 bb98 bb2 bb4 bb97
NMS algorithm set scores to zero for redundant bboxes
bb3 bb1 bb98 bb2 bb4 bb97
class = max_index(scores for bb97);
score = max(scores for bb97);
Score > 0
no
skip bbox
yes
draw bbox with class color


bb1 bb2 bb3 bb4
bb97 bb98
20x1
20x1
......
Set zero
if score < thresh1 (0.2)
bb1 bb2 bb3 bb4
bb97 bb98
0
0
0
......
Sort descending
bb3 bb1 bb98
bb2 bb4 bb97
0 0 0
......
NMS algorithm set
scores to zero for
redundant bboxes
bb3 bb1 bb98
bb2 bb4 bb97
0
0 0 0
......