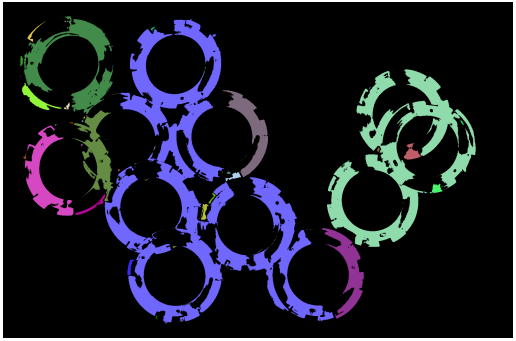
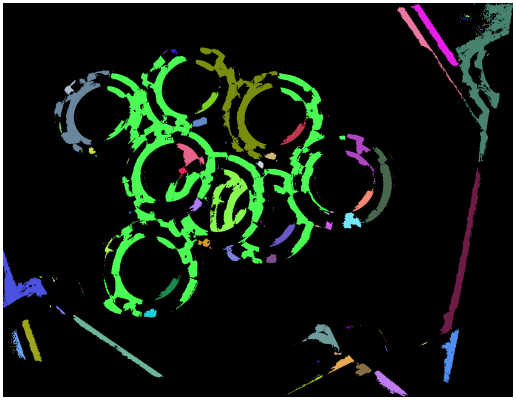
# Object Localization | Region Proposals

Lukáš Gajdošech

11.11.2020 - 23.11.2020

# Segmentation Result





# Segmentation vs. Bounding Box Localization

Localization $\subset$ Segmentation

# Similar CNN architectures

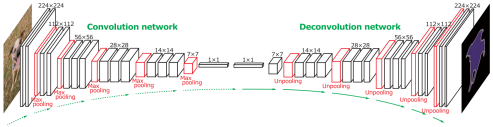



Figure: output: $WxH$ segmentation map

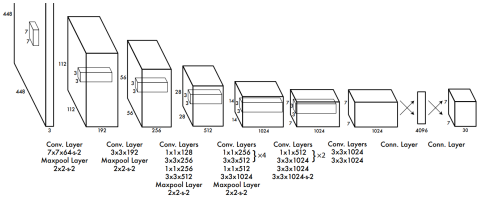
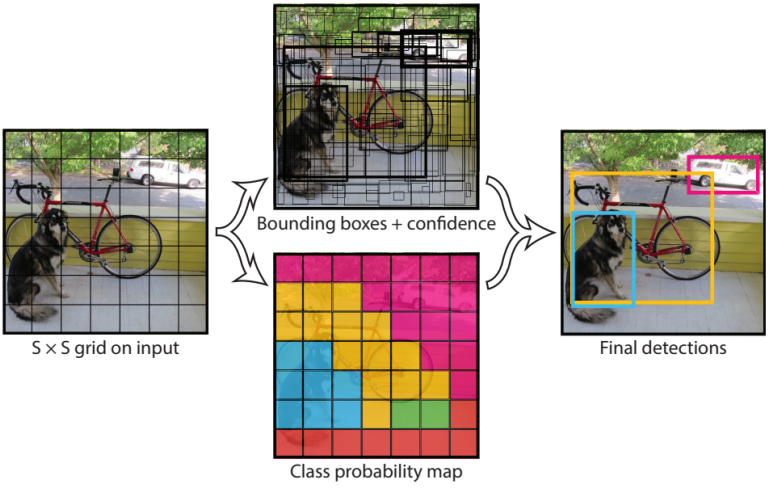



Figure: output: $SxSx(B * 5 + C)$ tensor

# Papers

- YOLO: You only look once https://arxiv.org/pdf/1506.02640.pdf https://arxiv.org/pdf/1804.02767.pdf
- Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks https://arxiv.org/pdf/1506.01497.pdf
- RetinaNET: Focal Loss for Dense Object Detection https://arxiv.org/pdf/1708.02002.pdf

# YOLO I


Bounding boxes + confidence
S × S grid on input
Final detections
Class probability map

# YOLO II

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 $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$. 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.

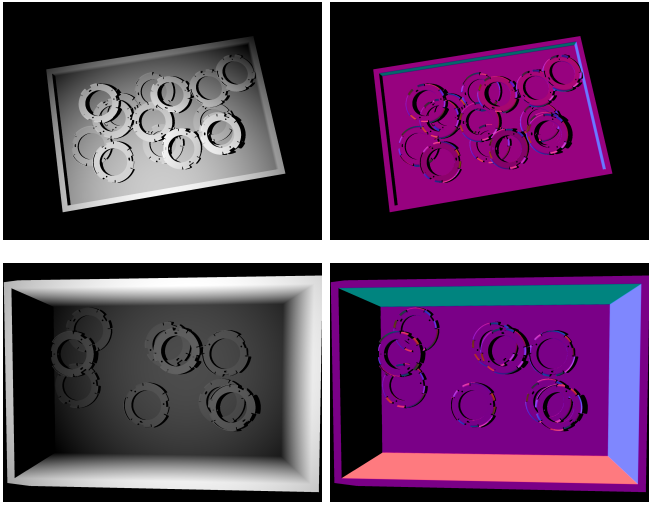
Each bounding box consists of 5 predictions: $x$, $y$, $w$, $h$,

# YOLO III

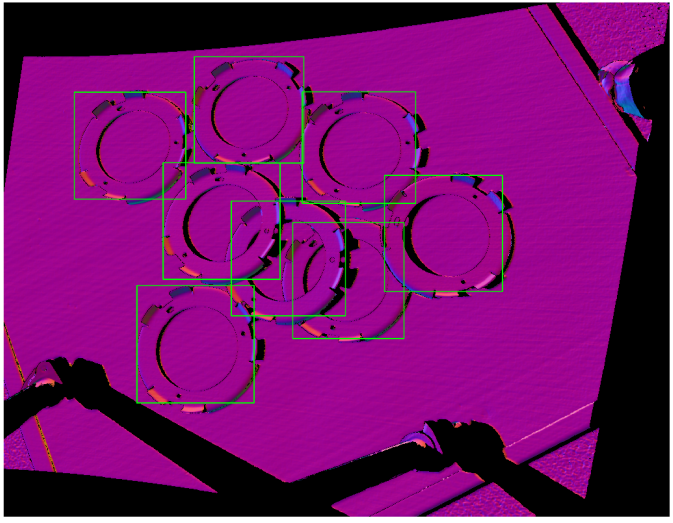
loss function:

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

# BinGen data

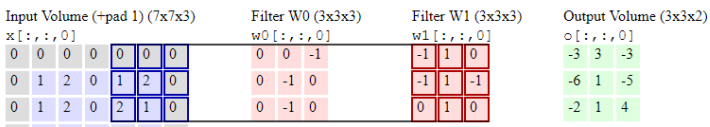


# Inference on Real Scan (RetinaNet)



# 15 pages of document...

**Convolution Demo**. Below is a running demo of a CONV layer. Since 3D volumes are hard to visualize, all the volumes (the input volume (in blue), the weight volumes (in red), the output volume (in green)) are visualized with each depth slice stacked in rows. The input volume is of size $W_1 = 5, H_1 = 5, D_1 = 3$, and the CONV layer parameters are $K = 2, F = 3, S = 2, P = 1$. That is, we have two filters of size $3 \times 3$, and they are applied with a stride of 2. Therefore, the output volume size has spatial size (5 - 3 + 2)/2 + 1 = 3. Moreover, notice that a padding of $P = 1$ is applied to the input volume, making the outer border of the input volume zero. The visualization below iterates over the output activations (green), and shows that each element is computed by elementwise multiplying the highlighted input (blue) with the filter (red), summing it up, and then offsetting the result by the bias.




https://cs231n.github.io/convolutional-networks/