Presented by:

Jie Li
Takuto Sato

Original Slides from:
Classification + Localization

v.s.

Object Detection

CAT

v.s.

CAT, DOG, DUCK
Localization as Regression

**Input:** image

Only one object, simpler than detection

**Neural Net**

**Output:**
- Box coordinates (4 numbers)

**Correct output:**
- box coordinates (4 numbers)

**Loss:**
- L2 distance

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n
Sliding Window: Overfeat (DPM as well!)

In practice use many sliding window locations and multiple scales

Window positions + score maps

Box regression outputs

Final Predictions

Detection as Regression?

DOG, (x, y, w, h)
CAT, (x, y, w, h)
CAT, (x, y, w, h)
DUCK (x, y, w, h)

= 16 numbers
Detection as Regression?

DOG, (x, y, w, h)
CAT, (x, y, w, h)

= 8 numbers
Detection as Regression?

CAT, (x, y, w, h)
CAT, (x, y, w, h)
....
CAT (x, y, w, h)
= many numbers

Need variable sized outputs
Detection as Classification

CAT? NO

DOG? NO
Detection as Classification

CAT? YES!

DOG? NO
Detection as Classification

CAT? NO

DOG? NO
Detection as Classification

**Problem:** Need to test many positions and scales

**Solution:** If your classifier is fast enough, just do it
Detection as Classification

**Problem:** Need to test many positions and scales, and use a computationally demanding classifier (CNN)

**Solution:** Only look at a tiny subset of possible positions
Region Proposals

- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions
Region Proposals: Selective Search

Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes

## Region Proposals: Many other choices

<table>
<thead>
<tr>
<th>Method</th>
<th>Approach</th>
<th>Outputs Segments</th>
<th>Outputs Score</th>
<th>Control proposals</th>
<th>Time (sec.)</th>
<th>Repeatability</th>
<th>Recall Results</th>
<th>Detection Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bing [18]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.2</td>
<td>⋆⋆⋆</td>
<td>⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>CPMC [19]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>250</td>
<td>-</td>
<td>⋆⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>EdgeBoxes [20]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.3</td>
<td>⋆⋆</td>
<td>⋆⋆⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>Endres [21]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>100</td>
<td>-</td>
<td>⋆⋆⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>Geodesic [22]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>⋆</td>
<td>⋆⋆⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>MCG [23]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>30</td>
<td>⋆</td>
<td>⋆⋆⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>Objectness [24]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>Rahtu [25]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>RandomizedPrim’s [26]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>⋆</td>
<td>⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>Rantalankila [27]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>⋆⋆</td>
<td>-</td>
<td>⋆</td>
</tr>
<tr>
<td>Rigor [28]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>⋆</td>
<td>⋆⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>SelectiveSearch [29]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>⋆⋆⋆</td>
<td>⋆⋆⋆</td>
<td>⋆</td>
</tr>
<tr>
<td>Gaussian</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>⋆</td>
</tr>
<tr>
<td>SlidingWindow</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Superpixels</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Uniform</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Hosang et al, “What makes for effective detection proposals?”, PAMI 2015

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

comp150dl

Tufts University
Putting it together: R-CNN

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

Slide credit: Ross Girshick

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n
**Fast R-CNN (test time)**

- **Softmax classifier**
  - Linear + softmax
- **Linear**
- **Bounding-box regressors**
- **Fully-connected layers**
- **“Roi Pooling” (single-level SPP) layer**
- **“conv5” feature map of image**
- **Regions of Interest (RoIs) from a proposal method**
- **Forward whole image through ConvNet**

**R-CNN Problem #1:**
Slow at test-time due to independent forward passes of the CNN

**Solution:**
Share computation of convolutional layers between proposals for an image

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n
Faster R-CNN:

Insert a **Region Proposal Network (RPN)** after the last convolutional layer.

RPN trained to produce region proposals directly; no need for external region proposals!

After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN.


* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n
Summary:

How to frame detection problem?

Regression?
- OK for one or a fixed number of objects (localization with sliding windows)
- Sliding windows is slow (especially when the number of objects too large)
- Number of objects unknown?

Classification?
- Need region-based method to propose bounding boxes (~2000 boxes, R-CNN)
- Still kind of slow (especially considering real-time detection)
You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶


Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detectors on several datasets including PASCAL VOC and ImageNet.

Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 x 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model’s confidence.

Figure 1 shows the YOLO detection system. YOLO processes an image by resizing it to 448x448, running a single convolutional network on the image, and then thresholds the resulting detections by the model's confidence.

methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

We focus our object detection research on improving end-to-end systems like YOLO, which are fast but not yet superior to the best methods.
We split the image into a grid

S = 7

Total # of Cells: S x S = 49

Original Slides from:
Each cell predicts boxes and confidences: $P(\text{Object})$

$S = 7$

Total # of Cells:
$S \times S = 49$

Each cell predicts boxes and confidences: $P(\text{Object})$

$S = 7$

Total # of Cells:
$S \times S = 49$

$B = 2$

Original Slides from:
Each cell predicts boxes and confidences: $P(\text{Object})$

$S = 7$

Total # of Cells: $S \times S = 49$

$B = 2$

Each cell predicts boxes and confidences: $P(\text{Object})$

$S = 7$

Total # of Cells:
$S \times S = 49$

$B = 2$

Original Slides from:
Each cell predicts boxes and confidences: P(Object)

$S = 7$

Total # of Cells: $S \times S = 49$

$B = 2$

Each cell predicts boxes and confidences: \( P(\text{Object}) \)

- \( S = 7 \)
- Total # of Cells: \( S \times S = 49 \)
- \( B = 2 \)
- Total # of boxes: \( S \times S \times B = 98 \)

Each cell also predicts a class probability.
Each cell also predicts a class probability.
Conditioned on object: $P(\text{Car} \mid \text{Object})$

Bicycle

Car

Dog

Dining Table
Then we combine the box and class predictions.

Original Slides from:
Finally we do NMS and threshold detections
This parameterization fixes the output size

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

$7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30$ tensor = 1470 outputs

Thus we can train one neural network to be a whole detection pipeline
During training, match example to the right cell
During training, match example to the right cell

Adjust that cell’s class prediction

Dog = 1
Cat = 0
Bike = 0
...

Look at that cell’s predicted boxes

Find the best one, make it “responsible for that prediction.
Find the best one, make it “responsible for that prediction"
loss function:

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

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

\[
+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{i,j}^{\text{obj}} (C_i - \hat{C}_i)^2
\]

\[
+ \lambda_{\text{noobj}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{i,j}^{\text{noobj}} (C_i - \hat{C}_i)^2
\]

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

(3)
Strengths and Weaknesses (compared to the R-CNN family)

Strengths

- Fast
- Generalizes better (i.e., good at detecting objects in paintings)
- Less background false positives

Weaknesses

- Localization not as accurate
- Limited to B objects per grid (B=2 in paper/this presentation)
Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating $1 \times 1$ convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ($224 \times 224$ input image) and then double the resolution for detection.