YOLO: You Only Look Once
Unified Real-Time Object Detection
Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

Name: Ranjeet Ranjan Jha
For: Deep Learning Class (12/04/17)
Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to $448 \times 448$, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model’s confidence.
Classification $+$ Localization: Task

**Classification**: $C$ classes
- **Input**: Image
- **Output**: Class label
- **Evaluation metric**: Accuracy

**Localization**:
- **Input**: Image
- **Output**: Box in the image $(x, y, w, h)$
- **Evaluation metric**: Intersection over Union

**Classification $+$ Localization**: Do both
Computer Vision Tasks

- Classification
- Classification + Localization
- Object Detection
- Instance Segmentation
Nowadays State of the Art approach, are so architected:
Detection as Classification

CAT? YES!
DOG? NO
Detection as Classification

CAT? NO
DOG? NO
Detection as Regression?

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

= 8 numbers
Region Proposals: Selective Search

Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes

Putting it together: R-CNN


Slide credit: Ross Girshick
This complex pipeline means that:

- Slow Pipeline
- Single Pipelines Hard to Optimize
- Need Parallel Training for Components
WHAT’S NEW?

(In the architecture approach.)
Concepts

- Detection as Single Regression Problem
- Developed as Single Convolutional Network
- Reason Globally on the Entire Image

Easy & Fast
Unified Detection
Divide the image into a SxS grid.

If the center of an object fall into a grid cell, it will be the responsible for the object.

Each grid cell predict:

- B bounding boxes;
- B confidence scores as $C = \text{Pr}(\text{Obj}) \times \text{IOU}$;
- C cond. Class prob. as $P = \text{Pr}(\text{Class}_i | \text{Object})$;

Confidence Prediction is obtained as IOU of predicted box and any ground truth box.
We obtain the class-specific confidence score as:

$$\Pr(Class_i | Object) \times \Pr(Object) \times IOU = \Pr(Class_i) \times IOU$$
Design

Conv. Layer 7x7x64 s2
Maxpool Layer 2x2 s2

Conv. Layer 3x3x192
Maxpool Layer 2x2 s2

Conv. Layers 1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer 2x2 s2

Conv. Layers 1x1x256
3x3x512
1x1x512
3x3x1024
Maxpool Layer 2x2 s2

Conv. Layers 1x1x512
3x3x1024
1x1x512
3x3x1024

Conv. Layers 3x3x1024
3x3x1024

Conn. Layer

Conn. Layer
\[ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{\text{obj}} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \]

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

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

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

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

- Sum-squared error: equally weights errors in large boxes and small boxes
- Small deviations in large boxes matter less than in small boxes.
- To partially address this, square root of the bounding box width and height are used.
Parameters

To avoid overfitting dropout and extensive data augmentation are used.

Dropout layer with rate = 0.5 after the first connected layer.

For data augmentation, random scaling and translations of up to 20% of the original image size.

Randomaly adjust the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.
To avoid overfitting dropout and extensive data augmentation are used.

Dropout layer with rate = 0.5 after the first connected layer.

For data augmentation, random scaling and translations of up to 20% of the original image size.

Randomly adjust the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.
Limitations

- Struggle with Small Object.
- Struggle with Different aspects and ratios of objects.
- Loss function is an approximation.
- Loss function threats errors in different boxes ratio at the same.
EXPERIMENTS

(How performs?.)
### General Comparison

<table>
<thead>
<tr>
<th>Real-Time Detectors</th>
<th>Train</th>
<th>mAP</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>100Hz DPM [30]</td>
<td>2007</td>
<td>16.0</td>
<td>100</td>
</tr>
<tr>
<td><strong>Fast YOLO</strong></td>
<td>2007+2012</td>
<td>52.7</td>
<td><strong>155</strong></td>
</tr>
<tr>
<td><strong>YOLO</strong></td>
<td>2007+2012</td>
<td><strong>63.4</strong></td>
<td>45</td>
</tr>
<tr>
<td><strong>Less Than Real-Time</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fastest DPM [37]</td>
<td>2007</td>
<td>30.4</td>
<td>15</td>
</tr>
<tr>
<td>R-CNN Minus R [20]</td>
<td>2007</td>
<td>53.5</td>
<td>6</td>
</tr>
<tr>
<td><strong>Fast R-CNN [14]</strong></td>
<td>2007+2012</td>
<td>70.0</td>
<td>0.5</td>
</tr>
<tr>
<td>Faster R-CNN VGG-16 [27]</td>
<td>2007+2012</td>
<td>73.2</td>
<td>7</td>
</tr>
<tr>
<td>Faster R-CNN ZF [27]</td>
<td>2007+2012</td>
<td>62.1</td>
<td>18</td>
</tr>
</tbody>
</table>
# Fast R-CNN & YOLO

Using YOLO accuracy for Big object to avoid detection mistakes into Fast R-CNN:

<table>
<thead>
<tr>
<th></th>
<th>mAP</th>
<th>Combined</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast R-CNN</td>
<td>-</td>
<td>71.8</td>
<td>-</td>
</tr>
<tr>
<td>Fast R-CNN (2007 data)</td>
<td>66.9</td>
<td>72.4</td>
<td>.6</td>
</tr>
<tr>
<td>Fast R-CNN (VGG-M)</td>
<td>59.2</td>
<td>72.4</td>
<td>.6</td>
</tr>
<tr>
<td>Fast R-CNN (CaffeNet)</td>
<td>57.1</td>
<td>72.1</td>
<td>.3</td>
</tr>
<tr>
<td>YOLO</td>
<td>63.4</td>
<td>75.0</td>
<td>3.2</td>
</tr>
<tr>
<td>VOC 2012 test</td>
<td>mAP</td>
<td>aeroplane</td>
<td>bicycle</td>
</tr>
<tr>
<td>---------------</td>
<td>-----</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>MR_CNN_MORE_DATE [11]</td>
<td>73.9</td>
<td>85.5</td>
<td>82.9</td>
</tr>
<tr>
<td>HyperNet_VGG</td>
<td>71.4</td>
<td>84.2</td>
<td>78.5</td>
</tr>
<tr>
<td>HyperNet.SP</td>
<td>71.3</td>
<td>84.1</td>
<td>78.3</td>
</tr>
<tr>
<td><strong>Fast R-CNN + YOLO</strong></td>
<td>70.7</td>
<td>83.4</td>
<td>78.5</td>
</tr>
<tr>
<td>MR_CNN.S.CNN [11]</td>
<td>70.7</td>
<td>85.0</td>
<td>79.6</td>
</tr>
<tr>
<td>Faster R-CNN [27]</td>
<td>70.4</td>
<td>84.9</td>
<td>79.8</td>
</tr>
<tr>
<td>DEEP.ENS.COCA</td>
<td>70.1</td>
<td>84.0</td>
<td>79.4</td>
</tr>
<tr>
<td>NoC [28]</td>
<td>68.8</td>
<td>82.8</td>
<td>79.0</td>
</tr>
<tr>
<td>Fast R-CNN [14]</td>
<td>68.4</td>
<td>82.3</td>
<td>78.4</td>
</tr>
<tr>
<td>UMIC.FGS.STRUCT</td>
<td>66.4</td>
<td>82.9</td>
<td>76.1</td>
</tr>
<tr>
<td>NUS.NIN.C2000 [7]</td>
<td>63.8</td>
<td>80.2</td>
<td>73.8</td>
</tr>
<tr>
<td>BabyLearning [7]</td>
<td>63.2</td>
<td>78.0</td>
<td>74.2</td>
</tr>
<tr>
<td>NUS.NIN</td>
<td>62.4</td>
<td>77.9</td>
<td>73.1</td>
</tr>
<tr>
<td><strong>R-CNN VGG BB [13]</strong></td>
<td>62.4</td>
<td>79.6</td>
<td>72.7</td>
</tr>
<tr>
<td><strong>R-CNN VGG [13]</strong></td>
<td>59.2</td>
<td>76.8</td>
<td>70.9</td>
</tr>
<tr>
<td><strong>YOLO</strong></td>
<td>57.9</td>
<td>77.0</td>
<td>67.2</td>
</tr>
<tr>
<td>Feature Edit [32]</td>
<td>56.3</td>
<td>74.6</td>
<td>69.1</td>
</tr>
<tr>
<td>R-CNN BB [13]</td>
<td>53.3</td>
<td>71.8</td>
<td>65.8</td>
</tr>
<tr>
<td>SDS [16]</td>
<td>50.7</td>
<td>69.7</td>
<td>58.4</td>
</tr>
<tr>
<td>R-CNN [13]</td>
<td>49.6</td>
<td>68.1</td>
<td>63.8</td>
</tr>
</tbody>
</table>
YOLO: You Only Look Once
Detection as Regression

Divide image into $S \times S$ grid

Within each grid cell predict:
- B Boxes: 4 coordinates + confidence
- Class scores: $C$ numbers

Regression from image to
$7 \times 7 \times (5 \times B + C)$ tensor

Direct prediction using a CNN

Pros

- Trained on a loss function that directly corresponds to detection performance.
- The entire model is trained jointly.
- The fastest general-purpose object detector in the literature.
- At least detection at 45fps.
Object Detection code links:

**R-CNN**
(Caffe + MATLAB): [https://github.com/rbgirshick/rcnn](https://github.com/rbgirshick/rcnn)
Probably don’t use this; too slow

**Fast R-CNN**
(Caffe + MATLAB): [https://github.com/rbgirshick/fast-rcnn](https://github.com/rbgirshick/fast-rcnn)

**Faster R-CNN**
(Caffe + MATLAB): [https://github.com/ShaoqingRen/faster_rcnn](https://github.com/ShaoqingRen/faster_rcnn)
(Caffe + Python): [https://github.com/rbgirshick/py-faster-rcnn](https://github.com/rbgirshick/py-faster-rcnn)

**YOLO**
Maybe try this for projects?
References

THANKS !!!

QUESTIONS?