R-FCN: Object Detection via Region-based Fully Convolutional Networks

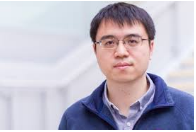
Kris Korrel Diede Rusticus

Master Artificial Intelligence
University of Amsterdam

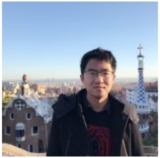
Computer Vision 2, 2017

Outline

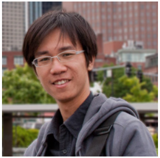
Microsoft Researchers - Paper: June 2016



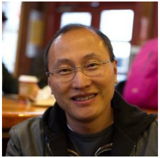
(a) Jian Sun



(b) Jifeng Dai



(c) Kaiming He



(d) Yi Li

R-FCN

Introducing R-FCN:

Region-based Fully Convolutional Network

- Shared, fully convolutional (like FCN)
- Translation variance incorporated

Previous work

AlexNet (2012), VGGNet (2015), ResNet (2016), etc.

- Designed for image classification
- Convolutional and pooling layers, followed by fully connected layers
- Used convolutional layers for translation invariance

R-CNN (2014)

- Designed for object detection
- RoIs proposed by region proposal algorithm (like Selective Search)
- Similar design, but RoI-wise

Fast R-CNN (2015)

- Extends on R-CNN
- One forward pass through convolutional layers followed by RoI-wise fully connected layers
- Introduces RoI pooling layer
- 0.3s test time excluding generation of region proposals

Faster R-CNN (2015)

- Extends on Fast R-CNN
- Introduces Region Proposal Network
- Trained alternatively with object detection network

Subdivided networks

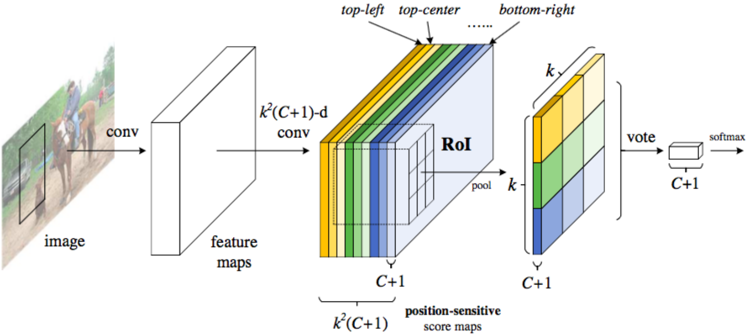
- Translation invariance needed for image classification
- Translation variance needed for object detection
- Fast and Faster R-CNN solve this by RoI pooling
- No shared computation in second subnetwork
- Can not take advantage of properties of FCNs like ResNet and GoogLeNet

Method

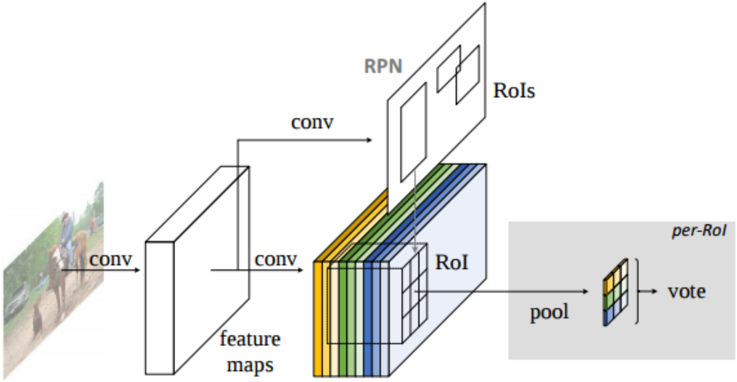
Overview:

- All learnable weights are fully convolutional
- Last layer produces a bank of position-sensitive score maps
- Ends with position-sensitive RoI pooling and voting, which is not learned
- Negligible RoI-wise computation
- RPN from Faster R-CNN is used for proposing RoIs

Architecture



Architecture



Network design

- ResNet-101 as backbone architecture
- Dimensionality reduction from 2048-d to 1024-d
- Last layer produces bank of $k^2(C+1)$ position-sensitive score maps

Position-sensitive pooling

For each RoI:

- Position-sensitive RoI pooling
  $K^2(C+1)$ score maps $\rightarrow$ $K^2(C+1)$ scores
- Voting
  $K^2(C+1)$ scores $\rightarrow$ $C+1$ scores
- Softmax responses

Bounding boxes

- Bounding boxes are proposed by RPN as introduced by Faster R-CNN
- R-FCN and RPN are trained alternatively
- Bounding box regression is done similarly to object detection
- bank of $4k^2$ score maps that predict (t_x, t_y, t_w, t_h)

Training

The loss function defined on each RoI is the summation of the **cross-entropy loss** and the **box regression loss**:

$$L(s, t_{x,y,w,h}) = \underbrace{Lcls(s_{c^*})}_{\text{classif. loss}} + \lambda \underbrace{[\overbrace{c^*}^{\text{ground truth label}} > 0]}_{\text{indicator (t/f)}} \underbrace{L_{reg}(t, \overbrace{t^*}^{\text{ground truth box}})}_{\text{bounding box regr. loss}}$$

Positive example $=$ IoU > 0.5

Online hard example mining

OHEM is easy to adopt:

- The RoIs are sorted by their losses, X RoIs with the **highest loss** are used for **backpropagation**
- Per-RoI computation is negligible so forward time is not affected by N
- In Fast R-CNN this doubles training time!

Default Settings

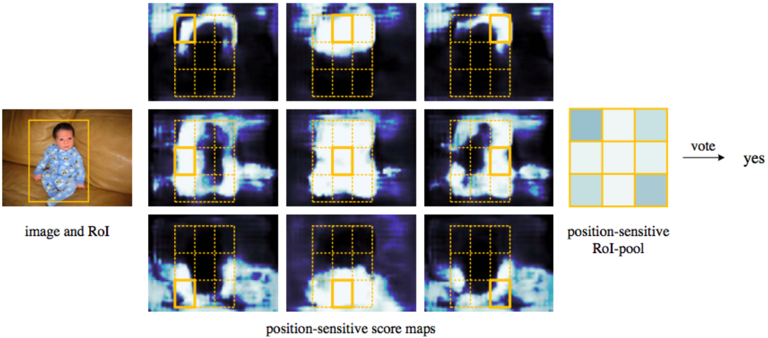
$$
\begin{aligned}
\text{weight decay} &= 0.0005 \\
\text{momentum} &= 0.9 \\
\text{single-scale training} &= \text{images are resized: shorter side is 600 px.} \\
\text{OHEM} &= \text{128 RoIs for backprop.} \\
\text{GPU} &= \text{1 RoI per GPU - 8 GPU's used} \\
\text{fine-tune learning rate} &= 0.001 \ / \ 0.0001
\end{aligned}
$$

Inference

1. Feature maps are computed
2. RPN proposes RoIs
3. R-FCN evaluates category scores and regresses bounding boxes
4. To compare with Faster R-CNN; also 300 RoIs are evaluated
5. Non-maximum suppression as post-processing

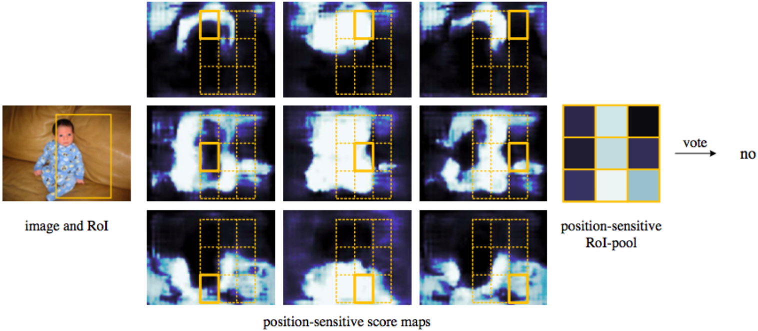
Visualisation

Person category: $(k \times k = 3 \times 3)$



Visualisation

Person category: $(k \times k = 3 \times 3)$



Dataset

PASCAL VOC

- 20 object categories
- trained on VOC 2007 + 2012 trainval set, evaluated on VOC 2007 test set
- Performance measure: mAP

Results 1/4

Comparison: other **Fully Convolutional** Strategies

method	RoI output size ($k \times k$)	mAP on VOC 07 (%)
naïve Faster R-CNN	1×1 7×7	61.7 68.9
class-specific RPN	-	67.6
R-FCN (w/o position-sensitivity)	1×1	*fail*
R-FCN	3×3 7×7	75.5 **76.6**

Results 2/4

Comparison: Faster R-CNN using **ResNet-101**

	depth of per-RoI subnetwork	training w/ OHEM?	train time (sec/img)	test time (sec/img)	mAP (%) on VOC07
Faster R-CNN	10		1.2	0.42	76.4
R-FCN	0		0.45	0.17	76.6
Faster R-CNN	10	✓(300 RoIs)	1.5	0.42	79.3
R-FCN	0	✓(300 RoIs)	0.45	0.17	**79.5**
Faster R-CNN	10	✓(2000 RoIs)	2.9	0.42	*N/A*
R-FCN	0	✓(2000 RoIs)	0.46	0.17	79.3

Results 3/4

Comparison: Faster R-CNN using **ResNet-101**
Trained on MS COCO and finetuned with PASCAL VOC

	training data	mAP (%)	test time (sec/img)
Faster R-CNN [9]	07+12	76.4	0.42
Faster R-CNN +++ [9]	07+12+COCO	**85.6**	3.36
R-FCN	07+12	79.5	0.17
R-FCN multi-sc train	07+12	80.5	0.17
R-FCN multi-sc train	07+12+COCO	**83.6**	0.17

Results 4/4

Comparison: Faster R-CNN using **ResNet-101**
Trained on MS COCO and finetuned with PASCAL VOC

	training data	mAP (%)	test time (sec/img)
Faster R-CNN [9]	07++12	73.8	0.42
Faster R-CNN +++ [9]	07++12+COCO	**83.8**	3.36
R-FCN multi-sc train	07++12	$77.6^{\dagger}$	0.17
R-FCN multi-sc train	07++12+COCO	$\mathbf{82.0}^{\ddagger}$	0.17

Further results

Impact of depth

- Increased accuracy from depth 50 to 101
- Saturated accuracy at a depth of 152

Impact of region proposals

- Good generality of the method because competitive performances with other region proposal methods

	training data	test data	RPN [18]	SS [27]	EB [28]
R-FCN	07+12	07	**79.5**	77.2	77.8

Dataset / Settings

MC COCO

- 80 object categories
- 80 k train set, 40k val set, 20k test-dev set
- Some different training settings

Results

Comparison: Faster R-CNN using ResNet-101 using the MC COCO dataset

	training data	test data	AP@0.5	AP	AP small	AP medium	AP large	test time (sec/img)
Faster R-CNN [9]	train	val	48.4	27.2	6.6	28.6	45.0	0.42
R-FCN	train	val	48.9	27.6	8.9	30.5	42.0	0.17
R-FCN multi-sc train	train	val	49.1	27.8	8.8	30.8	42.2	0.17
Faster R-CNN +++ [9]	trainval	test-dev	**55.7**	**34.9**	15.6	38.7	50.9	3.36
R-FCN	trainval	test-dev	51.5	29.2	10.3	32.4	43.3	0.17
R-FCN multi-sc train	trainval	test-dev	51.9	29.9	10.8	32.8	45.0	0.17
R-FCN multi-sc train, test	trainval	test-dev	**53.2**	**31.5**	14.3	35.5	44.2	1.00

Limitations

- Benefit of dimensionality reduction not further explained
- Not explained why they use single scale training by default.
- Not clear what $k \times k$ means for Faster R-CNN.
- Missing analysis on hyperparameter k

Conclusion

1. Presented Region-based Fully Convolutional Networks (R-FCN)
2. Simple, accurate and efficient object detector
3. Adopts state-of-the-art image classification backbones (ResNet)
4. Comparable accuracy to Faster R-CNN, but much faster

Future Work

Body-parts based scoring maps

- Bulat, A., & Tzimiropoulos, G. (2016, October). Human pose estimation via convolutional part heatmap regression. In European Conference on Computer Vision (pp. 717-732). Springer International Publishing. ISO 690

References

- Shrivastava, A., Gupta, A., & Girshick, R. (2016). Training region-based object detectors with online hard example mining. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 761-769). ISO 690
- Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In Advances in Neural Information Processing Systems (pp. 379-387). ISO 690
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).