# Fast, Faster and Mask-RCNN

Fuxin Li

CS637

# Composite Modeling: Object Proposals

- Previous lecture talked about object proposals
- Original CPMC/Selective Search generates category-free proposals
  - Based only on low-level features: color, edge, sizes
  - Good for bigger objects (PASCAL VOC) with distinct edges
  - Not good when edges are not distinct, or color distribution too complex
- “Deep Learned” proposal generation
  - Generation in conjunction with prediction
  - Use “higher-level” semantic features in deep learning

# Faster R-CNN

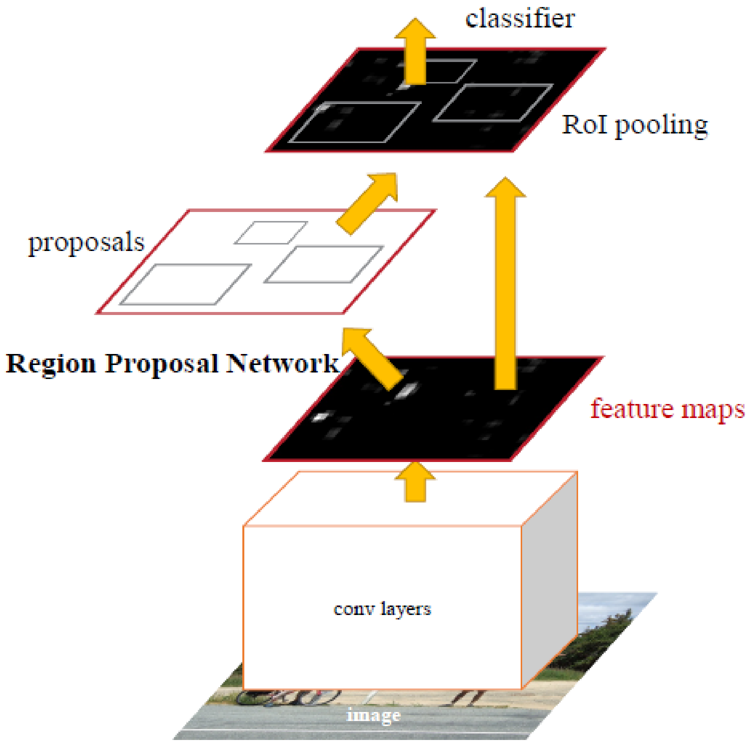



Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

# Region Proposal Networks

- Predict coordinates relative to *anchor* boxes
  - Pre-defined anchor boxes (3 scales, 3 aspect ratios)
  - Convolutional network: each location gets k predicted boxes

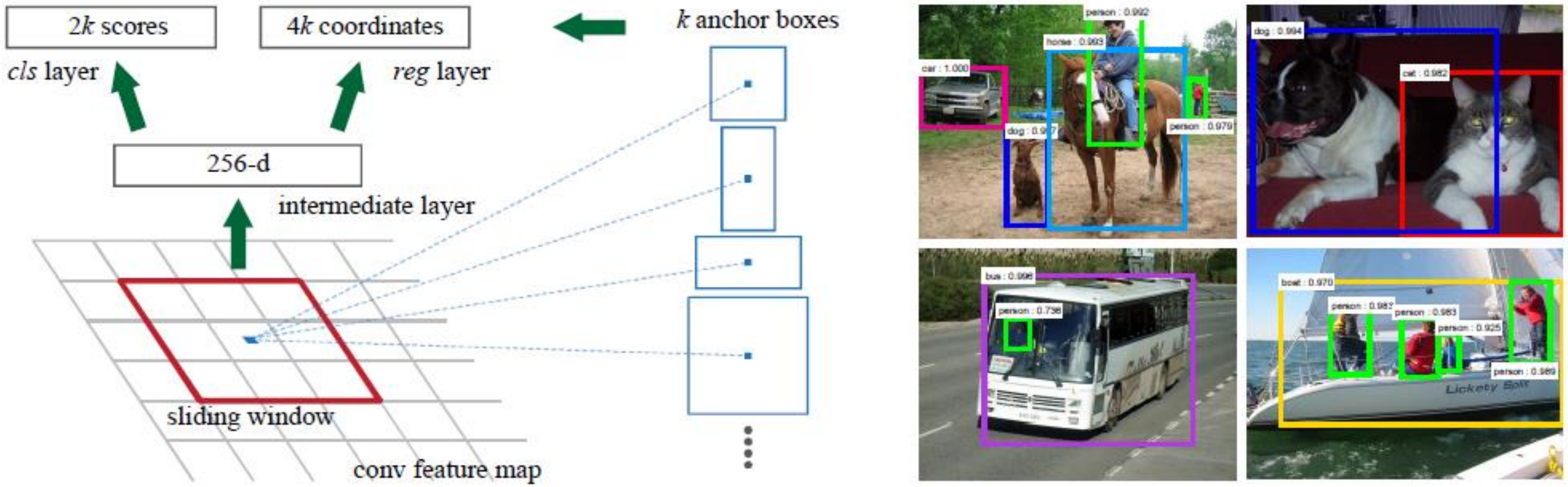



Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

# Cls layer and reg layer

- Cls layer: "Objectness"
  - Rank proposals on whether they look like an object (of training categories) or not
  - 2-class softmax prediction
- Reg layer: Bounding box coordinate offsets

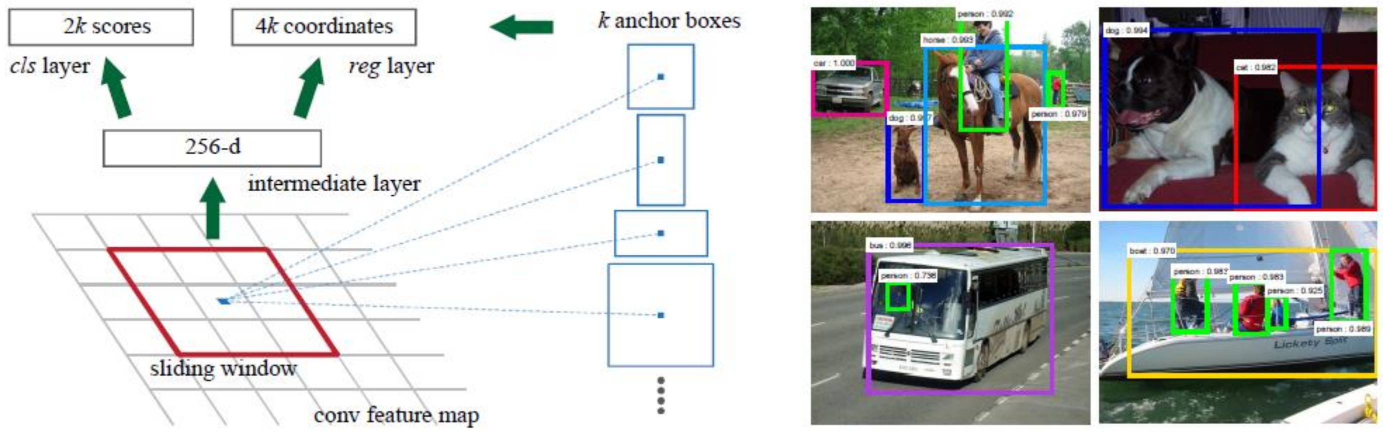



Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

# Loss Function

- Classification:
  - Two-class softmax
- Coordinate Regression:
  - Huber L1-loss on coordinates

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*). \tag{1}$$

Here, $i$ is the index of an anchor in a mini-batch and $p_i$ is the predicted probability of anchor $i$ being an object. The ground-truth label $p_i^*$ is 1 if the anchor is positive, and is 0 if the anchor is negative. $t_i$ is a vector representing the 4 parameterized coordinates of the predicted bounding box, and $t_i^*$ is that of the ground-truth box associated with a positive anchor. The classification loss $L_{cls}$ is log loss over two classes (object *vs.* not object). For the regression loss, we use $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where $R$ is the robust loss function (smooth $L_1$) defined in [2]. The term $p_i^* L_{reg}$ means the regression loss is activated only for positive anchors ($p_i^* = 1$) and is disabled otherwise ($p_i^* = 0$).

# After Having Box Proposals

- ROI Pooling
  - Divide a box into WxH (7x7) regions and generate 49 features for each filter
  - Max-pooling within each region
- Run multi-class classification on the ROI
- Run bounding box refinement regressor on the ROI

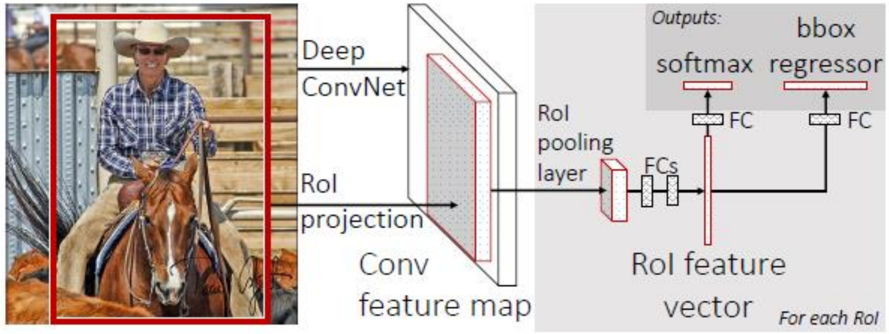



Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

# Training

- Loss functions:
  - RPN objectness loss
  - RPN bounding box regression loss
  - Fast R-CNN classification loss
  - Fast R-CNN bounding box regression loss
- Alternate training between RPN and fast R-CNN

# Experiment Results of Faster R-CNN

Table 8: Detection results of Faster R-CNN on PASCAL VOC 2007 test set using **different settings of anchors**. The network is VGG-16. The training data is VOC 2007 trainval. The default setting of using 3 scales and 3 aspect ratios (69.9%) is the same as that in Table 3.

| settings | anchor scales | aspect ratios | mAP (%) |
|---|---|---|---|
| 1 scale, 1 ratio | $128^2$ | 1:1 | 65.8 |
| | $256^2$ | 1:1 | 66.7 |
| 1 scale, 3 ratios | $128^2$ | {2:1, 1:1, 1:2} | 68.8 |
| | $256^2$ | {2:1, 1:1, 1:2} | 67.9 |
| 3 scales, 1 ratio | $\{128^2, 256^2, 512^2\}$ | 1:1 | **69.8** |
| 3 scales, 3 ratios | $\{128^2, 256^2, 512^2\}$ | {2:1, 1:1, 1:2} | **69.9** |

Table 9: Detection results of Faster R-CNN on PASCAL VOC 2007 test set using **different values of** $\lambda$ in Equation (1). The network is VGG-16. The training data is VOC 2007 trainval. The default setting of using $\lambda = 10$ (69.9%) is the same as that in Table 3.

| $\lambda$ | 0.1 | 1 | 10 | 100 |
|---|---|---|---|---|
| mAP (%) | 67.2 | 68.9 | 69.9 | 69.1 |

# How good are the proposals?

- Compared against a traditional proposal approach selective search

Table 6: Results on PASCAL VOC 2007 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000. RPN* denotes the unsharing feature version.

| method | # box | data | mAP | areo | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | person | plant | sheep | sofa | train | tv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SS | 2000 | 07 | 66.9 | 74.5 | 78.3 | 69.2 | 53.2 | 36.6 | 77.3 | 78.2 | 82.0 | 40.7 | 72.7 | 67.9 | 79.6 | 79.2 | 73.0 | 69.0 | 30.1 | 65.4 | 70.2 | 75.8 | 65.8 |
| SS | 2000 | 07+12 | 70.0 | 77.0 | 78.1 | 69.3 | 59.4 | 38.3 | 81.6 | 78.6 | 86.7 | 42.8 | 78.8 | 68.9 | 84.7 | 82.0 | 76.6 | 69.9 | 31.8 | 70.1 | 74.8 | 80.4 | 70.4 |
| RPN* | 300 | 07 | 68.5 | 74.1 | 77.2 | 67.7 | 53.9 | 51.0 | 75.1 | 79.2 | 78.9 | 50.7 | 78.0 | 61.1 | 79.1 | 81.9 | 72.2 | 75.9 | 37.2 | 71.4 | 62.5 | 77.4 | 66.4 |
| RPN | 300 | 07 | 69.9 | 70.0 | 80.6 | 70.1 | 57.3 | 49.9 | 78.2 | 80.4 | 82.0 | 52.2 | 75.3 | 67.2 | 80.3 | 79.8 | 75.0 | 76.3 | 39.1 | 68.3 | 67.3 | 81.1 | 67.6 |
| RPN | 300 | 07+12 | 73.2 | 76.5 | 79.0 | 70.9 | 65.5 | 52.1 | 83.1 | 84.7 | 86.4 | 52.0 | 81.9 | 65.7 | 84.8 | 84.6 | 77.5 | 76.7 | 38.8 | 73.6 | 73.9 | 83.0 | 72.6 |
| RPN | 300 | COCO+07+12 | **78.8** | **84.3** | **82.0** | **77.7** | **68.9** | **65.7** | **88.1** | **88.4** | **88.9** | **63.6** | **86.3** | **70.8** | **85.9** | **87.6** | **80.1** | **82.3** | **53.6** | **80.4** | **75.8** | **86.6** | **78.9** |

Table 7: Results on PASCAL VOC 2012 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000.

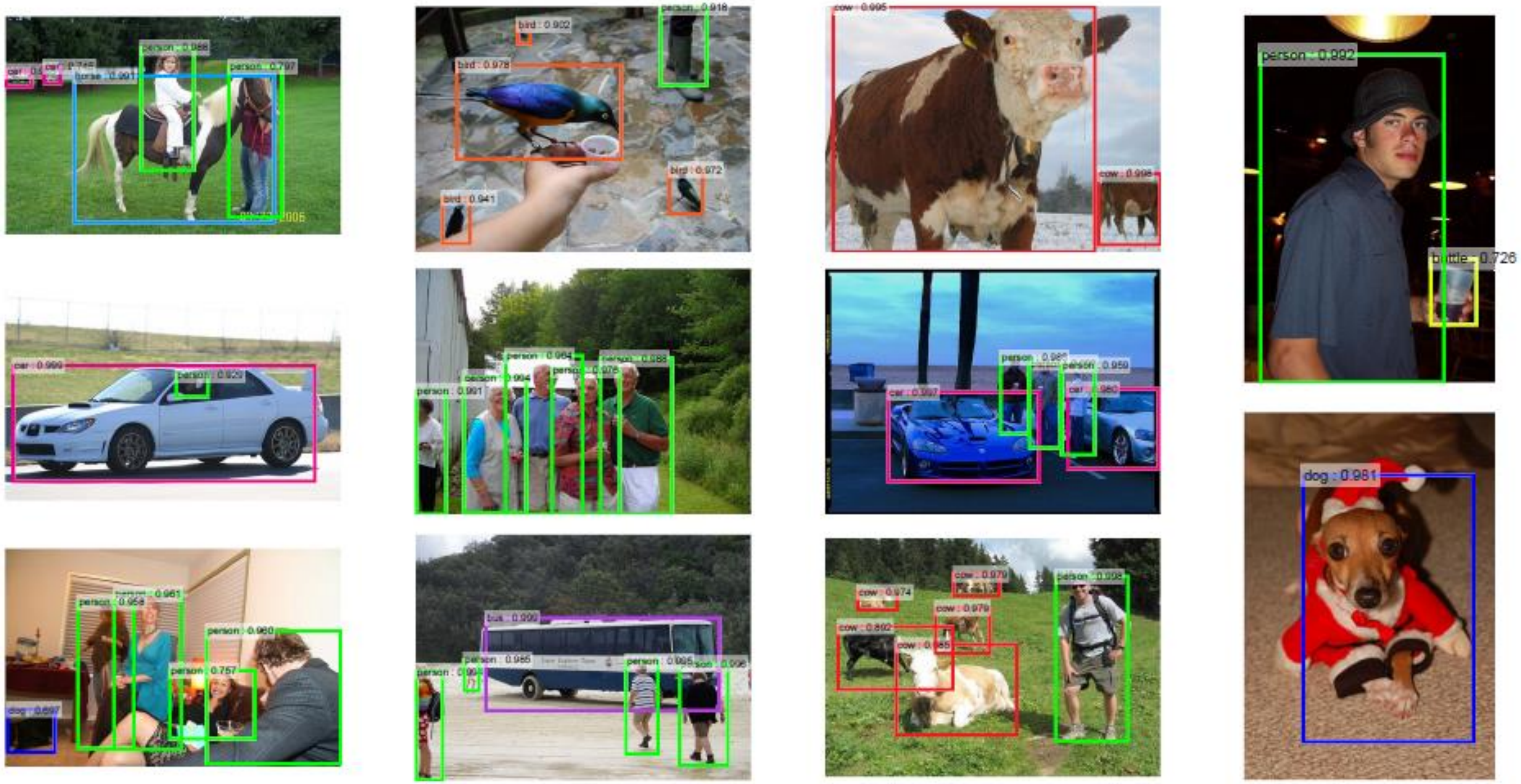
| method | # box | data | mAP | areo | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | person | plant | sheep | sofa | train | tv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SS | 2000 | 12 | 65.7 | 80.3 | 74.7 | 66.9 | 46.9 | 37.7 | 73.9 | 68.6 | 87.7 | 41.7 | 71.1 | 51.1 | 86.0 | 77.8 | 79.8 | 69.8 | 32.1 | 65.5 | 63.8 | 76.4 | 61.7 |
| SS | 2000 | 07++12 | 68.4 | 82.3 | 78.4 | 70.8 | 52.3 | 38.7 | 77.8 | 71.6 | 89.3 | 44.2 | 73.0 | 55.0 | **87.5** | 80.5 | 80.8 | 72.0 | 35.1 | 68.3 | **65.7** | 80.4 | 64.2 |
| RPN | 300 | 12 | 67.0 | 82.3 | 76.4 | 71.0 | 48.4 | 45.2 | 72.1 | 72.3 | 87.3 | 42.2 | 73.7 | 50.0 | 86.8 | 78.7 | 78.4 | 77.4 | 34.5 | 70.1 | 57.1 | 77.1 | 58.9 |
| RPN | 300 | 07++12 | 70.4 | 84.9 | 79.8 | 74.3 | 53.9 | 49.8 | 77.5 | 75.9 | 88.5 | 45.6 | 77.1 | 55.3 | 86.9 | 81.7 | 80.9 | 79.6 | 40.1 | 72.6 | 60.9 | 81.2 | 61.5 |
| RPN | 300 | COCO+07++12 | **75.9** | **87.4** | **83.6** | **76.8** | **62.9** | **59.6** | **81.9** | **82.0** | **91.3** | **54.9** | **82.6** | **59.0** | **89.0** | **85.5** | **84.7** | **84.1** | **52.2** | **78.9** | 65.5 | **85.4** | **70.2** |

# How good are the proposals?

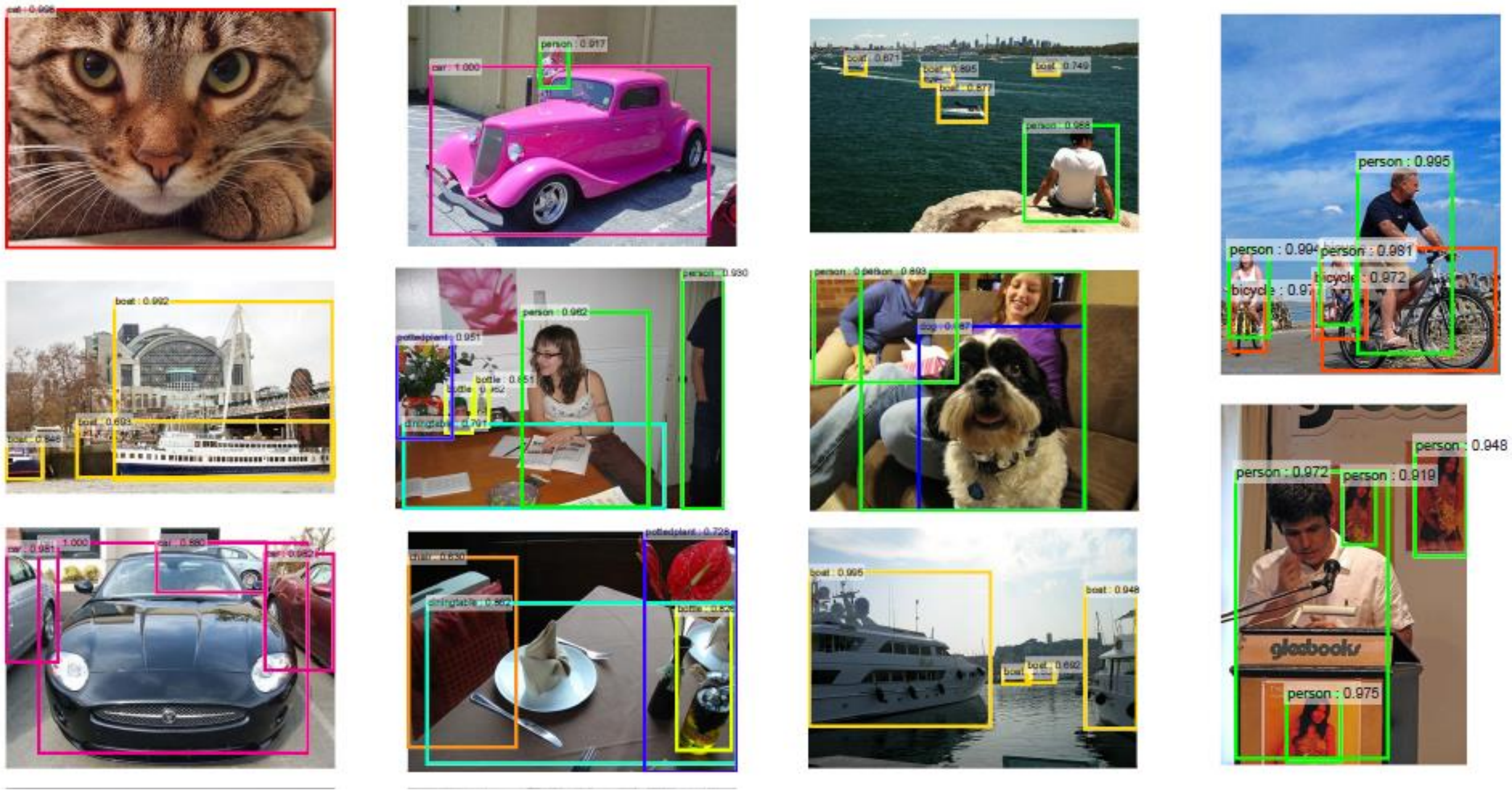
Table 11: Object detection results (%) on the **MS COCO** dataset. The model is VGG-16.

| method | proposals | training data | COCO val | | COCO test-dev | |
|---|---|---|---|---|---|---|
| | | | mAP@.5 | mAP@[.5, .95] | mAP@.5 | mAP@[.5, .95] |
| Fast R-CNN [2] | SS, 2000 | COCO train | - | - | 35.9 | 19.7 |
| Fast R-CNN [impl. in this paper] | SS, 2000 | COCO train | 38.6 | 18.9 | 39.3 | 19.3 |
| Faster R-CNN | RPN, 300 | COCO train | 41.5 | 21.2 | 42.1 | 21.5 |
| Faster R-CNN | RPN, 300 | COCO trainval | - | - | **42.7** | **21.9** |

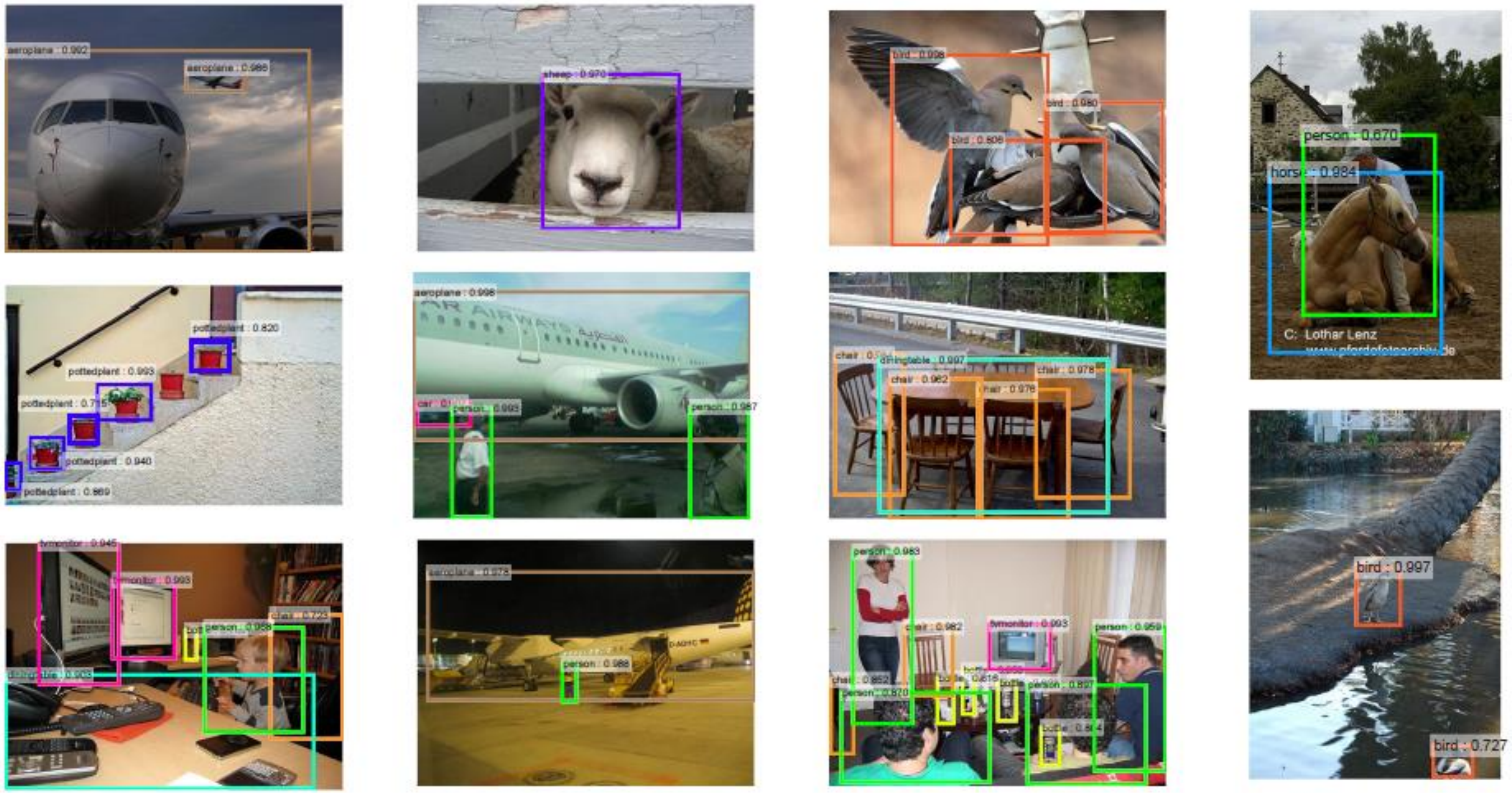
# How good does the results look like?



# How good does the results look like?



# How good does the results look like?



# Mask R-CNN

- Add segmentation mask prediction for each ROI
- Predict one mask per category
  - 0.6% improvement vs. predicting just one mask

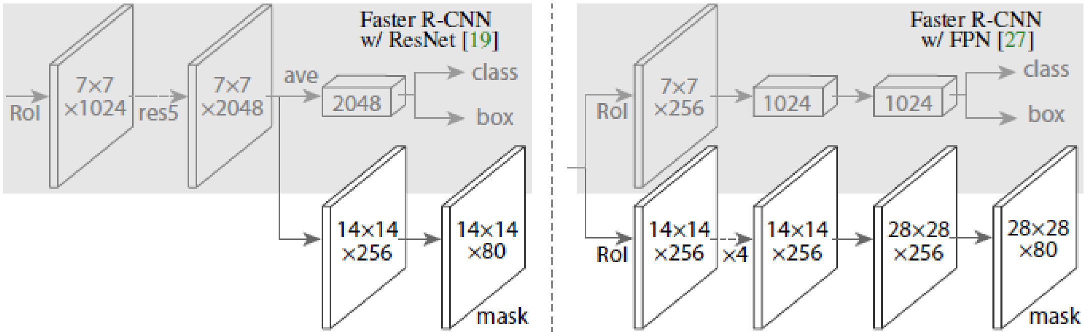



Figure 3. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or *fc* layers as can be inferred from context (conv preserves spatial dimension while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [30] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.

# ROI Align

- It is proposed to change ROIPool to ROIAlign, namely, no longer perform rounding when computing the ROIPool region
  - [X/16] => X/16
  - Use bilinear interpolation to compute pixel values

| | align? | bilinear? | agg. | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|---|
| *RoIPool* [12] | | | max | 26.9 | 48.8 | 26.4 |
| *RoIWarp* [10] | | ✓ | max | 27.2 | 49.2 | 27.1 |
| | | ✓ | ave | 27.1 | 48.9 | 27.1 |
| *RoIAlign* | ✓ | ✓ | max | **30.2** | **51.0** | **31.8** |
| | ✓ | ✓ | ave | **30.3** | **51.2** | **31.5** |

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and $AP_{75}$ by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

# Binary vs. Multinomial Loss on the Mask

- Mask R-CNN uses a binary loss on the mask

| | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| *softmax* | 24.8 | 44.1 | 25.1 |
| *sigmoid* | **30.3** | **51.2** | **31.5** |
| | *+5.5* | *+7.1* | *+6.4* |

(b) **Multinomial *vs.* Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

# Results on Detection

| | backbone | $AP^{bb}$ | $AP^{bb}_{50}$ | $AP^{bb}_{75}$ | $AP^{bb}_{S}$ | $AP^{bb}_{M}$ | $AP^{bb}_{L}$ |
|---|---|---|---|---|---|---|---|
| Faster R-CNN+++ [19] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [27] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI [21] | Inception-ResNet-v2 [37] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM [36] | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | **52.1** |
| Faster R-CNN, RoIAlign | ResNet-101-FPN | 37.3 | 59.6 | 40.3 | 19.8 | 40.2 | 48.8 |
| **Mask R-CNN** | ResNet-101-FPN | 38.2 | 60.3 | 41.7 | 20.1 | 41.1 | 50.2 |
| **Mask R-CNN** | ResNeXt-101-FPN | **39.8** | **62.3** | **43.4** | **22.1** | **43.2** | 51.2 |

Table 3. **Object detection** *single-model* results (bounding box AP), *vs.* state-of-the-art on `test-dev`. Mask R-CNN using ResNet-101-FPN outperforms the base variants of all previous state-of-the-art models (the mask output is ignored in these experiments). The gains of Mask R-CNN over [27] come from using RoIAlign (+1.1 $AP^{bb}$), multitask training (+0.9 $AP^{bb}$), and ResNeXt-101 (+1.6 $AP^{bb}$).

# Results on Instance Segmentation (COCO+Cityscapes)

| | backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| MNC [10] | ResNet-101-C4 | 24.6 | 44.3 | 24.8 | 4.7 | 25.9 | 43.6 |
| FCIS [26] +OHEM | ResNet-101-C5-dilated | 29.2 | 49.5 | - | 7.1 | 31.3 | 50.0 |
| FCIS+++ [26] +OHEM | ResNet-101-C5-dilated | 33.6 | 54.5 | - | - | - | - |
| **Mask R-CNN** | ResNet-101-C4 | 33.1 | 54.9 | 34.8 | 12.1 | 35.6 | 51.1 |
| **Mask R-CNN** | ResNet-101-FPN | 35.7 | 58.0 | 37.8 | 15.5 | 38.1 | 52.4 |
| **Mask R-CNN** | ResNeXt-101-FPN | **37.1** | **60.0** | **39.4** | **16.9** | **39.9** | **53.5** |

Table 1. **Instance segmentation** *mask* AP on COCO `test-dev`. MNC [10] and FCIS [26] are the winners of the COCO 2015 and 2016 segmentation challenges, respectively. Without bells and whistles, Mask R-CNN outperforms the more complex FCIS+++, which includes multi-scale train/test, horizontal flip test, and OHEM [35]. All entries are *single-model* results.

| | training data | AP [`val`] | AP | $AP_{50}$ | person | rider | car | truck | bus | train | mcycle | bicycle |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| InstanceCut [23] | `fine` + `coarse` | 15.8 | 13.0 | 27.9 | 10.0 | 8.0 | 23.7 | 14.0 | 19.5 | 15.2 | 9.3 | 4.7 |
| DWT [4] | `fine` | 19.8 | 15.6 | 30.0 | 15.1 | 11.7 | 32.9 | 17.1 | 20.4 | 15.0 | 7.9 | 4.9 |
| SAIS [17] | `fine` | - | 17.4 | 36.7 | 14.6 | 12.9 | 35.7 | 16.0 | 23.2 | 19.0 | 10.3 | 7.8 |
| DIN [3] | `fine` + `coarse` | - | 20.0 | 38.8 | 16.5 | 16.7 | 25.7 | 20.6 | 30.0 | 23.4 | 17.1 | 10.1 |
| Mask R-CNN | `fine` | 31.5 | 26.2 | 49.9 | 30.5 | 23.7 | 46.9 | 22.8 | 32.2 | 18.6 | 19.1 | 16.0 |
| Mask R-CNN | `fine` + COCO | **36.4** | **32.0** | **58.1** | **34.8** | **27.0** | **49.1** | **30.1** | **40.9** | **30.9** | **24.1** | **18.7** |

Table 7. Results on Cityscapes `val` ('AP [`val`]' column) and `test` (remaining columns) sets. Our method uses ResNet-50-FPN.

# Qualitative Results on Instance Segmentation

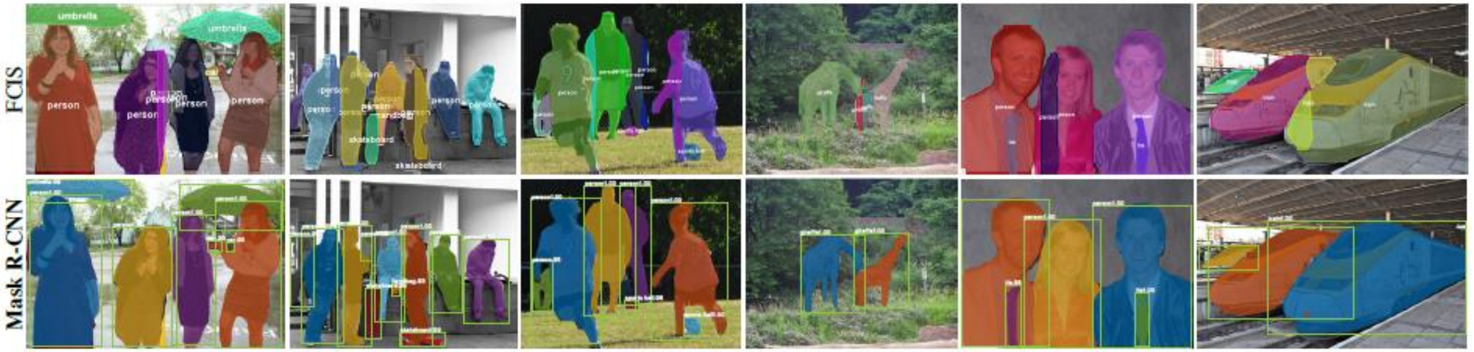



Figure 5. FCIS+++ [26] (top) *vs.* Mask R-CNN (bottom, ResNet-101-FPN). FCIS exhibits systematic artifacts on overlapping objects.

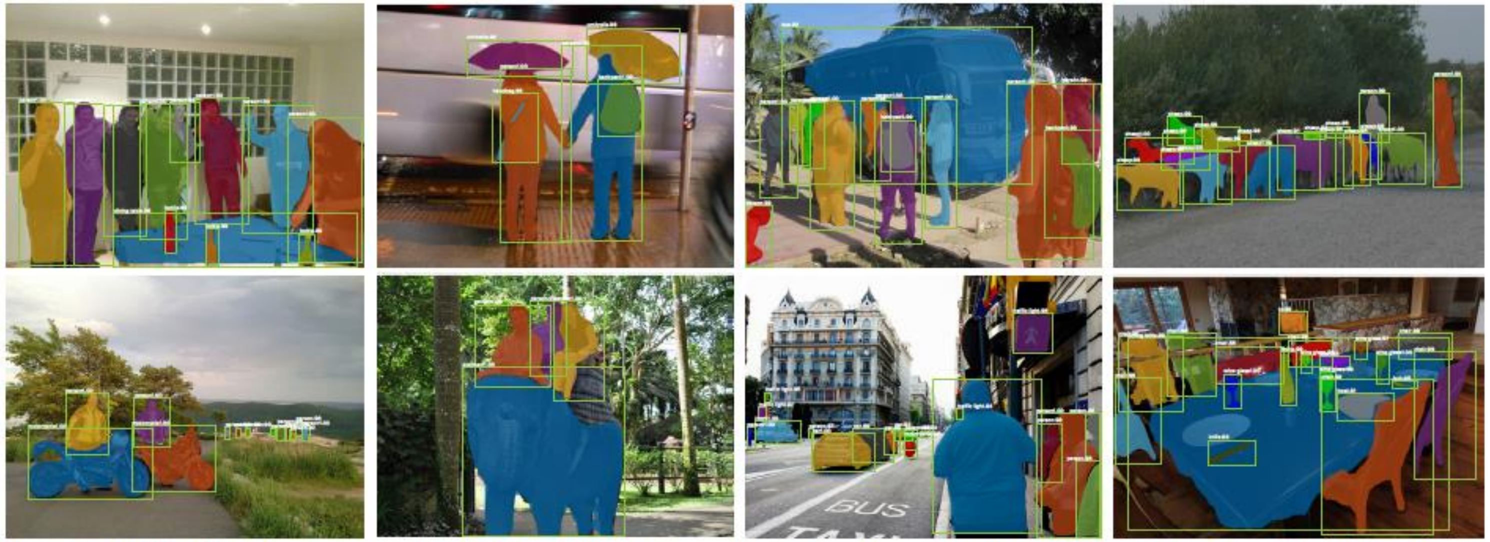


Figure 2. Mask R-CNN results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

# More Qualitative Results

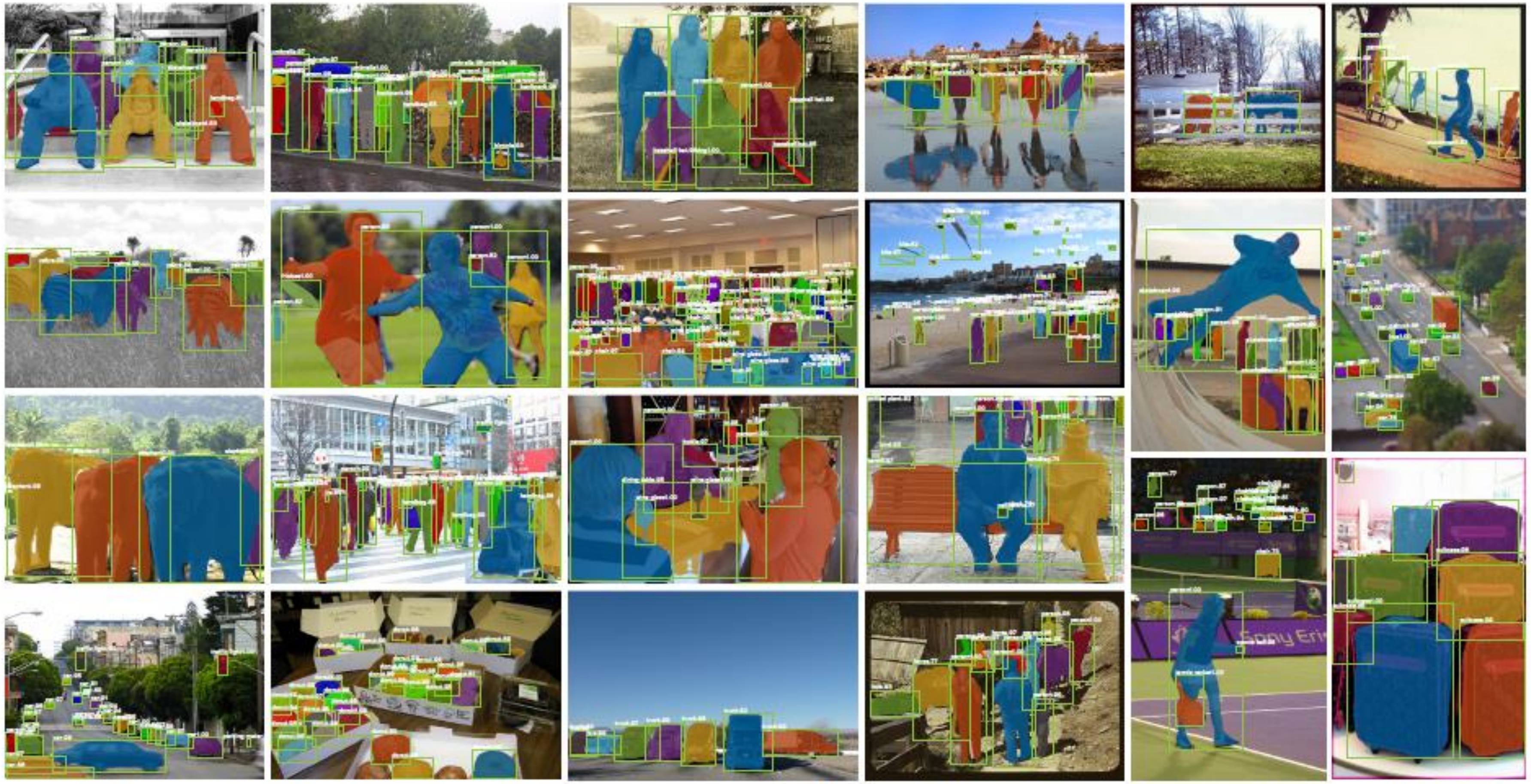


Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

# Qualitative results on Cityscapes

Figure 7. Mask R-CNN results on Cityscapes `test` (32.0 AP). The bottom-right image shows a failure prediction.