Mask R-CNN

Kaiming He  Georgia Gkioxari  Piotr Dollár  Ross Girshick
Facebook AI Research (FAIR)

Abstract

We present a conceptually simple, flexible, and general framework for object instance segmentation. Our approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps. Moreover, Mask R-CNN is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework. We show top results in all three tracks of the COCO suite of challenges, including instance segmentation, bounding-box object detection, and person keypoint detection. Without bells and whistles, Mask R-CNN outperforms all existing, single-model entries on every task, including the COCO 2016 challenge winners. We hope our simple and effective approach will serve as a solid baseline and help ease future research in instance-level recognition. Code has been made available at: https://github.com/facebookresearch/Detectron.

1. Introduction

The vision community has rapidly improved object detection and semantic segmentation results over a short period of time. In large part, these advances have been driven by powerful baseline systems, such as the Fast/Faster R-CNN [12, 36] and Fully Convolutional Network (FCN) [30] frameworks for object detection and semantic segmentation, respectively. These methods are conceptually intuitive and offer flexibility and robustness, together with fast training and inference time. Our goal in this work is to develop a comparably enabling framework for instance segmentation.

Instance segmentation is challenging because it requires the correct detection of all objects in an image while also precisely segmenting each instance. It therefore combines elements from the classical computer vision tasks of object detection, where the goal is to classify individual objects and localize each using a bounding box, and semantic segmentation, where the goal is to classify each pixel into a fixed set of categories without differentiating object instances. Given this, one might expect a complex method is required to achieve good results. However, we show that a surprisingly simple, flexible, and fast system can surpass prior state-of-the-art instance segmentation results.

Our method, called Mask R-CNN, extends Faster R-CNN [36] by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression (Figure 1). The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner. Mask R-CNN is simple to implement and train given the Faster R-CNN framework, which facilitates a wide range of flexible architecture designs. Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

In principle Mask R-CNN is an intuitive extension of Faster R-CNN, yet constructing the mask branch properly is critical for good results. Most importantly, Faster R-CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. This is most evident in how RoIPool [18, 12], the de facto core operation for attending to instances, performs coarse spatial quantization for feature extraction. To fix the misalignment, we propose a simple, quantization-free layer, called RoIAlign, that faithfully preserves exact spatial locations. Despite being

1Following common terminology, we use object detection to denote detection via bounding boxes, not masks, and semantic segmentation to denote per-pixel classification without differentiating instances. Yet we note that instance segmentation is both semantic and a form of detection.
a seemingly minor change, RoIAlign has a large impact: it improves mask accuracy by relative 10% to 50%, showing bigger gains under stricter localization metrics. Second, we found it essential to decouple mask and class prediction: we predict a binary mask for each class independently, without competition among classes, and rely on the network’s RoI classification branch to predict the category. In contrast, FCNs usually perform per-pixel multi-class categorization, which couples segmentation and classification, and based on our experiments works poorly for instance segmentation.

Without bells and whistles, Mask R-CNN surpasses all previous state-of-the-art single-model results on the COCO instance segmentation task [28], including the heavily-engineered entries from the 2016 competition winner. As a by-product, our method also excels on the COCO object detection task. In ablation experiments, we evaluate multiple basic instantiations, which allows us to demonstrate its robustness and analyze the effects of core factors.

Our models can run at about 200ms per frame on a GPU, and training on COCO takes one to two days on a single 8-GPU machine. We believe the fast train and test speeds, together with the framework’s flexibility and accuracy, will benefit and ease future research on instance segmentation.

Finally, we showcase the generality of our framework via the task of human pose estimation on the COCO keypoint dataset [28]. By viewing each keypoint as a one-hot binary mask, with minimal modification Mask R-CNN can be applied to detect instance-specific poses. Mask R-CNN surpasses the winner of the 2016 COCO keypoint competition, and at the same time runs at 5 fps. Mask R-CNN, therefore, can be seen more broadly as a flexible framework for instance-level recognition and can be readily extended to more complex tasks.

We have released code to facilitate future research.

2. Related Work

R-CNN: The Region-based CNN (R-CNN) approach [13] to bounding-box object detection is to attend to a manageable number of candidate object regions [42, 20] and evaluate convolutional networks [25, 24] independently on each RoI. R-CNN was extended [18, 12] to allow attending to RoIs on feature maps using RoIPool, leading to fast speed and better accuracy. Faster R-CNN [36] advanced this stream by learning the attention mechanism with a Region Proposal Network (RPN). Faster R-CNN is flexible and robust to many follow-up improvements (e.g., [38, 27, 21]), and is the current leading framework in several benchmarks.

Instance Segmentation: Driven by the effectiveness of R-CNN, many approaches to instance segmentation are based on segment proposals. Earlier methods [13, 15, 16, 9] resorted to bottom-up segments [42, 2]. DeepMask [33] and following works [34, 8] learn to propose segment candidates, which are then classified by Fast R-CNN. In these methods, segmentation precedes recognition, which is slow and less accurate. Likewise, Dai et al. [10] proposed a complex multiple-stage cascade that predicts segment proposals from bounding-box proposals, followed by classification. Instead, our method is based on parallel prediction of masks and class labels, which is simpler and more flexible.

Most recently, Li et al. [26] combined the segment proposal system in [8] and object detection system in [11] for “fully convolutional instance segmentation” (FCIS). The common idea in [8, 11, 26] is to predict a set of position-sensitive output channels fully convolutionally. These channels simultaneously address object classes, boxes, and masks, making the system fast. But FCIS exhibits systematic errors on overlapping instances and creates spurious edges (Figure 6), showing that it is challenged by the fundamental difficulties of segmenting instances.
Another family of solutions \cite{23, 4, 3, 29} to instance segmentation are driven by the success of semantic segmentation. Starting from per-pixel classification results (e.g., FCN outputs), these methods attempt to cut the pixels of the same category into different instances. In contrast to the segmentation-first strategy of these methods, Mask R-CNN is based on an instance-first strategy. We expect a deeper incorporation of both strategies will be studied in the future.

3. Mask R-CNN

Mask R-CNN is conceptually simple: Faster R-CNN has two outputs for each candidate object, a class label and a bounding-box offset; to this we add a third branch that outputs the object mask. Mask R-CNN is thus a natural and intuitive idea. But the additional mask output is distinct from the class and box outputs, requiring extraction of much finer spatial layout of an object. Next, we introduce the key elements of Mask R-CNN, including pixel-to-pixel alignment, which is the main missing piece of Fast/Faster R-CNN.

Faster R-CNN: We begin by briefly reviewing the Faster R-CNN detector \cite{36}. Faster R-CNN consists of two stages. The first stage, called a Region Proposal Network (RPN), proposes candidate object bounding boxes. The second stage, which is in essence Fast R-CNN \cite{12}, extracts features using RoIPool from each candidate box and performs classification and bounding-box regression. The features used by both stages can be shared for faster inference. We refer readers to \cite{21} for latest, comprehensive comparisons between Faster R-CNN and other frameworks.

Mask R-CNN: Mask R-CNN adopts the same two-stage procedure, with an identical first stage (which is RPN). In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI. This is in contrast to most recent systems, where classification depends on mask predictions (e.g. \cite{33, 10, 26}). Our approach follows the spirit of Fast R-CNN \cite{12} that applies bounding-box classification and regression in parallel (which turned out to largely simplify the multi-stage pipeline of original R-CNN \cite{13}).

Formally, during training, we define a multi-task loss on each sampled RoI as \( L = L_{cls} + L_{box} + L_{mask} \). The classification loss \( L_{cls} \) and bounding-box loss \( L_{box} \) are identical as those defined in \cite{12}. The mask branch has a \( K m^2 \)-dimensional output for each RoI, which encodes \( K \) binary masks of resolution \( m \times m \), one for each of the \( K \) classes. To this we apply a per-pixel sigmoid, and define \( L_{mask} \) as the average binary cross-entropy loss. For an RoI associated with ground-truth class \( k \), \( L_{mask} \) is only defined on the \( k \)-th mask (other mask outputs do not contribute to the loss).

Our definition of \( L_{mask} \) allows the network to generate masks for every class without competition among classes; we rely on the dedicated classification branch to predict the class label used to select the output mask. This decouples mask and class prediction. This is different from common practice when applying FCNs \cite{30} to semantic segmentation, which typically uses a per-pixel softmax and a multinomial cross-entropy loss. In that case, masks across classes compete; in our case, with a per-pixel sigmoid and a binary loss, they do not. We show by experiments that this formulation is key for good instance segmentation results.

Mask Representation: A mask encodes an input object’s spatial layout. Thus, unlike class labels or box offsets that are inevitably collapsed into short output vectors by fully-connected (fc) layers, extracting the spatial structure of masks can be addressed naturally by the pixel-to-pixel correspondence provided by convolutions.

Specifically, we predict an \( m \times m \) mask from each RoI using an FCN \cite{30}. This allows each layer in the mask branch to maintain the explicit \( m \times m \) object spatial layout without collapsing it into a vector representation that lacks spatial dimensions. Unlike previous methods that resort to fc layers for mask prediction \cite{33, 34, 10}, our fully convolutional representation requires fewer parameters, and is more accurate as demonstrated by experiments.

This pixel-to-pixel behavior requires our RoI features, which themselves are small feature maps, to be well aligned to faithfully preserve the explicit per-pixel spatial correspondence. This motivated us to develop the following RoIAlign layer that plays a key role in mask prediction.

RoIAlign: RoIPool \cite{12} is a standard operation for extracting a small feature map (e.g., \( 7 \times 7 \)) from each RoI. RoIPool first quantizes a floating-number RoI to the discrete granularity of the feature map, this quantized RoI is then subdivided into spatial bins which are themselves quantized, and finally feature values covered by each bin are aggregated (usually by max pooling). Quantization is performed, e.g., on a continuous coordinate \( x \) by computing \( \lfloor x/16 \rfloor \), where 16 is a feature map stride and \( \lfloor \cdot \rfloor \) is rounding; likewise, quantization is performed when dividing into bins (e.g., \( 7 \times 7 \)). These quantizations introduce misalignments between the RoI and the extracted features. While this may not impact classification, which is robust to small translations, it has a large negative effect on predicting pixel-accurate masks.

To address this, we propose an RoIAlign layer that removes the harsh quantization of RoIPool, properly aligning the extracted features with the input. Our proposed change is simple: we avoid any quantization of the RoI boundaries.
or bins (i.e., we use $x/16$ instead of $[x/16]$). We use bi-linear interpolation [22] to compute the exact values of the input features at four regularly sampled locations in each RoI bin, and aggregate the result (using max or average), see Figure 3 for details. We note that the results are not sensitive to the exact sampling locations, or how many points are sampled, as long as no quantization is performed.

RoIAlign leads to large improvements as we show in §4.2. We also compare to the RoI Warp operation proposed in [10]. Unlike RoIAlign, RoI Warp overlooked the alignment issue and was implemented in [10] as quantizing RoI just like RoIPool. So even though RoI Warp also adopts bi-linear resampling motivated by [22], it performs on par with RoIPool as shown by experiments (more details in Table 2c), demonstrating the crucial role of alignment.

**Network Architecture:** To demonstrate the generality of our approach, we instantiate Mask R-CNN with multiple architectures. For clarity, we differentiate between: (i) the convolutional backbone architecture used for feature extraction over an entire image, and (ii) the network head for bounding-box recognition (classification and regression) and mask prediction that is applied separately to each RoI.

We denote the backbone architecture using the nomenclature `network-depth-features`. We evaluate ResNet [19] and ResNeXt [45] networks of depth 50 or 101 layers. The original implementation of Faster R-CNN with ResNets [19] extracted features from the final convolutional layer of the 4-th stage, which we call C4. This backbone with ResNet-50, for example, is denoted by ResNet-50-C4. This is a common choice used in [19, 10, 21, 39].

We also explore another more effective backbone recently proposed by Lin et al. [27], called a Feature Pyramid Network (FPN). FPN uses a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. Faster R-CNN with an FPN backbone extracts RoI features from different levels of the feature pyramid according to their scale, but otherwise the rest of the approach is similar to vanilla ResNet. Using a ResNet-FPN backbone for feature extraction with Mask R-CNN gives excellent gains in both accuracy and speed. For further details on FPN, we refer readers to [27].

For the network head we closely follow architectures presented in previous work to which we add a fully convolutional mask prediction branch. Specifically, we extend the Faster R-CNN box heads from the ResNet [19] and FPN [27] papers. Details are shown in Figure 4. The head on the ResNet-C4 backbone includes the 5-th stage of ResNet (namely, the 9-layer ‘res5’ [19]), which is compute-intensive. For FPN, the backbone already includes res5 and thus allows for a more efficient head that uses fewer filters.

We note that our mask branches have a straightforward structure. More complex designs have the potential to improve performance but are not the focus of this work.

**3.1. Implementation Details**

We set hyper-parameters following existing Fast/Faster R-CNN work [12, 36, 27]. Although these decisions were made for object detection in original papers [12, 36, 27], we found our instance segmentation system is robust to them.

**Training:** As in Fast R-CNN, an RoI is considered positive if it has IoU with a ground-truth box of at least 0.5 and negative otherwise. The mask loss $L_{mask}$ is defined only on positive RoIs. The mask target is the intersection between an RoI and its associated ground-truth mask.

We adopt image-centric training [12]. Images are resized such that their scale (shorter edge) is 800 pixels [27]. Each mini-batch has 2 images per GPU and each image has $N$ sampled RoIs, with a ratio of 1:3 of positive to negatives [12]. $N$ is 64 for the C4 backbone (as in [12, 36]) and 512 for FPN (as in [27]). We train on 8 GPUs (so effective mini-batch size is 16) for 160k iterations, with a learning rate of 0.02 which is decreased by 10 at the 120k iteration. We use a weight decay of 0.0001 and momentum of 0.9. With ResNeXt [45], we train with 1 image per GPU and the same number of iterations, with a starting learning rate of 0.01.

The RPN anchors span 5 scales and 3 aspect ratios, following [27]. For convenient ablation, RPN is trained separately and does not share features with Mask R-CNN, unless specified. For every entry in this paper, RPN and Mask R-CNN have the same backbones and so they are shareable.

**Inference:** At test time, the proposal number is 300 for the C4 backbone (as in [36]) and 1000 for FPN (as in [27]). We run the box prediction branch on these proposals, followed by non-maximum suppression [14]. The mask branch is then applied to the highest scoring 100 detection boxes. Although this differs from the parallel computation used in training, it speeds up inference and improves accuracy (due to the use of fewer, more accurate RoIs). The mask branch
can predict $K$ masks per RoI, but we only use the $k$-th mask, where $k$ is the predicted class by the classification branch. The $m \times m$ floating-number mask output is then resized to the RoI size, and binarized at a threshold of 0.5.

Note that since we only compute masks on the top 100 detection boxes, Mask R-CNN adds a small overhead to its Faster R-CNN counterpart (e.g., ~20% on typical models).

4. Experiments: Instance Segmentation

We perform a thorough comparison of Mask R-CNN to the state of the art along with comprehensive ablations on the COCO dataset [28]. We report the standard COCO metrics including AP (averaged over IoU thresholds), AP$_{0.5}$, AP$_{0.75}$, and AP$_S$, AP$_M$, AP$_L$ (AP at different scales). Unless noted, AP is evaluating using mask IoU. As in previous work [5, 27], we train using the union of 80k train images and a 35k subset of val images (trainval35k), and report ablations on the remaining 5k val images (minival). We also report results on test-dev [28].

4.1. Main Results

We compare Mask R-CNN to the state-of-the-art methods in instance segmentation in Table 1. All instantiations of our model outperform baseline variants of previous state-of-the-art models. This includes MNC [10] and FCIS [26], 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 [38]. All entries are single-model results.

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 [38]. All entries are single-model results.

<table>
<thead>
<tr>
<th>backbone</th>
<th>AP</th>
<th>AP$_{0.5}$</th>
<th>AP$_{0.75}$</th>
<th>AP$_S$</th>
<th>AP$_M$</th>
<th>AP$_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNC [10]</td>
<td>24.6</td>
<td>44.3</td>
<td>24.8</td>
<td>4.7</td>
<td>25.9</td>
<td>43.6</td>
</tr>
<tr>
<td>FCIS [26] +OHEM</td>
<td>20.2</td>
<td>49.5</td>
<td>-</td>
<td>7.1</td>
<td>31.3</td>
<td>50.0</td>
</tr>
<tr>
<td>FCIS+++ [26] +OHEM</td>
<td>33.6</td>
<td>54.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>33.1</td>
<td>54.9</td>
<td>34.8</td>
<td>12.1</td>
<td>35.6</td>
<td>51.1</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>35.7</td>
<td>58.0</td>
<td>37.8</td>
<td>15.5</td>
<td>38.1</td>
<td>52.4</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>37.1</td>
<td>60.0</td>
<td>39.4</td>
<td>16.9</td>
<td>39.9</td>
<td>53.5</td>
</tr>
</tbody>
</table>

Figure 5. 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).
Table 2. Ablations. We train on trainval35k, test on minival, and report mask AP unless otherwise noted.

4.2. Ablation Experiments

We run a number of ablations to analyze Mask R-CNN. Results are shown in Table 2 and discussed in detail next.

Architecture: Table 2a shows Mask R-CNN with various backbones. It benefits from deeper networks (50 vs. 101) and advanced designs including FPN and ResNeXt. We note that not all frameworks automatically benefit from deeper or advanced networks (see benchmarking in [21]).

Multinomial vs. Independent Masks: Mask R-CNN decouples mask and class prediction: as the existing box branch predicts the class label, we generate a mask for each class without competition among classes (i.e., per-class binary masks (sigmoid)) gives large gains over multinomial masks (softmax). (ResNet-50-C4): Decoupling via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

Class-Specific vs. Class-Agnostic Masks: Our default instantiation predicts class-specific masks, i.e., one \( m \times m \) mask per class. Interestingly, Mask R-CNN with class-agnostic masks (i.e., predicting a single \( m \times m \) output regardless of class) is nearly as effective: it has 29.7 mask AP vs. 30.3 for the class-specific counterpart on ResNet-50-C4. This further highlights the division of labor in our approach which largely decouples classification and segmentation.

RoIAlign: An evaluation of our proposed RoIAlign layer is shown in Table 2c. For this experiment we use the ResNet-50-C4 backbone, which has stride 16. RoIAlign improves AP by about 3 points over RoIPool, with much of the gain coming at high IoU (AP\(_{75}\)). RoIAlign is insensitive to max/average pool; we use average in the rest of the paper.

Additionally, we compare with RoIWarp proposed in MNC [10] that also adopt bilinear sampling. As discussed in §3, RoIWarp still quantizes the RoI, losing alignment with the input. As can be seen in Table 2c, RoIWarp performs on par with RoIPool and much worse than RoIAlign. This highlights that proper alignment is key.

We also evaluate RoIAlign with a ResNet-50-C5 backbone, which has an even larger stride of 32 pixels. We use the same head as in Figure 4 (right), as the res5 head is not applicable. Table 2d shows that RoIAlign improves mask AP by a massive 7.3 points, and mask AP\(_{75}\) by 10.5 points.

### Table 2: Ablations

<table>
<thead>
<tr>
<th>Architecture</th>
<th>AP</th>
<th>AP(_{50})</th>
<th>AP(_{75})</th>
<th>AP(_{bb})</th>
<th>AP(_{bb})</th>
<th>AP(_{bb})</th>
<th>mask branch</th>
<th>AP</th>
<th>AP(_{50})</th>
<th>AP(_{75})</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50-C4</td>
<td>30.3</td>
<td>51.2</td>
<td>31.5</td>
<td>24.8</td>
<td>44.1</td>
<td>25.1</td>
<td>RoIPool [12]</td>
<td>max</td>
<td>26.9</td>
<td>48.8</td>
</tr>
<tr>
<td>ResNet-101-C4</td>
<td>32.7</td>
<td>54.2</td>
<td>34.3</td>
<td>+5.5</td>
<td>+7.1</td>
<td>+6.4</td>
<td>RoWarp [10]</td>
<td>✓</td>
<td>27.2</td>
<td>49.2</td>
</tr>
<tr>
<td>ResNet-50-FPN</td>
<td>33.6</td>
<td>55.2</td>
<td>35.3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>RoIAlign</td>
<td>✓</td>
<td>✓</td>
<td>max</td>
</tr>
<tr>
<td>ResNet-101-FPN</td>
<td>35.4</td>
<td>57.3</td>
<td>37.5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>RoIAlign</td>
<td>✓</td>
<td>✓</td>
<td>ave</td>
</tr>
<tr>
<td>ResNeXt-101-FPN</td>
<td>36.7</td>
<td>59.5</td>
<td>38.9</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>RoIAlign</td>
<td>✓</td>
<td>✓</td>
<td>ave</td>
</tr>
</tbody>
</table>

(a) Backbone Architecture: Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.
Inception-ResNet-v2 \[41\] 
16.2 39.8 
13.5 38.1 52.0 
ResNet-101-FPN 
22.1 43.2 
ResNet-101-C4 
36.8 57.7 39.2 
34.7 55.5 36.7 
18.2 39.0 48.2 
ResNet-101-FPN 
19.8 40.2 48.8 
38.2 60.3 41.7 
ResNet-101-FPN 
36.2 59.1 39.0 
19.8 40.2 48.8 
Inception-ResNet-v2-TDM 
(50% relative improvement). Moreover, we note that with RoIAlign, using stride-32 C5 features (30.9 AP) is more accurate than using stride-16 C4 features (30.3 AP, Table 2c). RoIAlign largely resolves the long-standing challenge of using large-stride features for detection and segmentation.

Finally, RoIAlign shows a gain of 1.5 mask AP and 0.5 box AP when used with FPN, which has finer multi-level strides. For keypoint detection that requires finer alignment, RoIAlign shows large gains even with FPN (Table 6).

**Mask Branch:** Segmentation is a pixel-to-pixel task and we exploit the spatial layout of masks by using an FCN. In Table 2c, we compare multi-layer perceptrons (MLP) and FCNs, using a ResNet-50-FPN backbone. Using FCNs gives a 2.1 mask AP gain over MLPs. We note that we choose this backbone so that the conv layers of the FCN head are not pre-trained, for a fair comparison with MLP.

**4.3. Bounding Box Detection Results**

We compare Mask R-CNN to the state-of-the-art COCO bounding-box object detection in Table 3. For this result, even though the full Mask R-CNN model is trained, only the classification and box outputs are used at inference (the mask output is ignored). Mask R-CNN using ResNet-101-FPN outperforms the base variants of all previous state-of-the-art models, including the single-model variant of G-RMI \[21\], the winner of the COCO 2016 Detection Challenge. Using ResNeXt-101-FPN, Mask R-CNN further improves results, with a margin of 3.0 points box AP over the best previous single model entry from \[39\] (which used Inception-ResNet-v2-TDM).

As a further comparison, we trained a version of Mask R-CNN but without the mask branch, denoted as “Faster R-CNN, RoIAlign” in Table 3. This model performs better than the model presented in \[27\] due to RoIAlign. On the other hand, it is 0.9 points box AP lower than Mask R-CNN. This gap of Mask R-CNN on box detection is therefore due solely to the benefits of multi-task training.

Lastly, we note that Mask R-CNN attains a small gap between its mask and box AP, e.g., 2.7 points between 37.1 (mask, Table 1) and 39.8 (box, Table 3). This indicates that our approach largely closes the gap between object detection and the more challenging instance segmentation task.

**4.4. Timing**

**Inference:** We train a ResNet-101-FPN model that shares features between the RPN and Mask R-CNN stages, following the 4-step training of Faster R-CNN \[36\]. This model runs at 195ms per image on an Nvidia Tesla M40 GPU (plus 15ms CPU time resizing the outputs to the original resolution), and achieves statistically the same mask AP as the unshared one. We also report that the ResNet-101-C4 variant takes ~400ms as it has a heavier box head (Figure 4), so we do not recommend using the C4 variant in practice.

Although Mask R-CNN is fast, we note that our design is not optimized for speed, and better speed/accuracy trade-offs could be achieved \[21\], e.g., by varying image sizes and proposal numbers, which is beyond the scope of this paper.

**Training:** Mask R-CNN is also fast to train. Training with ResNet-50-FPN on COCO trainval35k takes 32 hours in our synchronized 8-GPU implementation (0.72s per 16-image mini-batch), and 44 hours with ResNet-101-FPN. In fact, fast prototyping can be completed in less than one day when training on the train set. We hope such rapid training will remove a major hurdle in this area and encourage more people to perform research on this challenging topic.

**5. Mask R-CNN for Human Pose Estimation**

Our framework can easily be extended to human pose estimation. We model a keypoint’s location as a one-hot mask, and adopt Mask R-CNN to predict \(K\) masks, one for each of \(K\) keypoint types (e.g., left shoulder, right elbow). This task helps demonstrate the flexibility of Mask R-CNN.

We note that minimal domain knowledge for human pose is exploited by our system, as the experiments are mainly to demonstrate the generality of the Mask R-CNN framework. We expect that domain knowledge (e.g., modeling structures \[6\]) will be complementary to our simple approach.

**Implementation Details:** We make minor modifications to the segmentation system when adapting it for keypoints. For each of the \(K\) keypoints of an instance, the training target is a one-hot \(m \times m\) binary mask where only a single pixel is labeled as foreground. During training, for each visible ground-truth keypoint, we minimize the cross-entropy loss over an \(m^2\)-way softmax output (which encourages a
We adopt the ResNet-FPN variant, and the keypoint head architecture is similar to that in Figure 4 (right). The keypoint head consists of a stack of eight $3 \times 3$ 512-d conv layers, followed by a deconv layer and $2 \times 2$ bilinear upsampling, producing an output resolution of $56 \times 56$. We found that a relatively high resolution output (compared to masks) is required for keypoint-level localization accuracy.

Models are trained on all COCO trainval35k images that contain annotated keypoints. To reduce overfitting, as this training set is smaller, we train using image scales randomly sampled from [640, 800] pixels; inference is on a single scale of 800 pixels. We train for 90k iterations, starting from a learning rate of 0.02 and reducing it by 10 at 60k and 80k iterations. We use bounding-box NMS with a threshold of 0.5. Other details are identical as in §3.1.

**Main Results and Ablations:** We evaluate the person keypoint AP ($\text{AP}_{kp}$) and experiment with a ResNet-50-FPN backbone; more backbones will be studied in the appendix. Table 4 shows that our result (62.7 $\text{AP}_{kp}$) is 0.9 points higher than the COCO 2016 keypoint detection winner [6] that uses a multi-stage processing pipeline (see caption of Table 4). Our method is considerably simpler and faster.

More importantly, we have a unified model that can simultaneously predict boxes, segments, and keypoints while running at 5 fps. Adding a segment branch (for the person category) improves the $\text{AP}_{kp}$ to 63.1 (Table 4) on test-dev. More ablations of multi-task learning on minival are in Table 5. Adding the mask branch to the box-only (i.e., Faster R-CNN) or keypoint-only versions consistently improves these tasks. However, adding the keypoint branch reduces the box/mask AP slightly, suggesting that while keypoint detection benefits from multitask training, it does not in turn help the other tasks. Nevertheless, learning all three tasks jointly enables a unified system to efficiently predict all outputs simultaneously (Figure 7).

We also investigate the effect of RoIAlign on keypoint detection (Table 6). Though this ResNet-50-FPN backbone has finer strides (e.g., 4 pixels on the finest level), RoIAlign still shows significant improvement over RoIPool and increases $\text{AP}_{kp}$ by 4.4 points. This is because keypoint detections are more sensitive to localization accuracy. This again indicates that alignment is essential for pixel-level localization, including masks and keypoints.

Given the effectiveness of Mask R-CNN for extracting object bounding boxes, masks, and keypoints, we expect it to be an effective framework for other instance-level tasks.

---

**Figure 7.** Keypoint detection results on COCO test using Mask R-CNN (ResNet-50-FPN), with person segmentation masks predicted from the same model. This model has a keypoint AP of 63.1 and runs at 5 fps.

**Table 4.** Keypoint detection AP on COCO test-dev. Ours is a single model (ResNet-50-FPN) that runs at 5 fps. CMU-Pose+++ [6] is the 2016 competition winner that uses multi-scale testing, post-processing with CPM [44], and filtering with an object detector, adding a cumulative threshold of 0.5. Other details are identical as in Table 6. Increasing that while keypoint detection benefits from multitask learning, it does not in turn help the other tasks. Nevertheless, learning all three tasks jointly enables a unified system to efficiently predict all outputs simultaneously (Figure 7).

<table>
<thead>
<tr>
<th></th>
<th>$\text{AP}_{kp}$</th>
<th>$\text{AP}_{kp, 50}$</th>
<th>$\text{AP}_{kp, 75}$</th>
<th>$\text{AP}_{kp, M}$</th>
<th>$\text{AP}_{kp, L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMU-Pose+++ [6]</td>
<td>61.8</td>
<td>84.9</td>
<td>67.5</td>
<td>57.1</td>
<td>68.2</td>
</tr>
<tr>
<td>G-RMI [32]†</td>
<td>62.4</td>
<td>84.0</td>
<td>68.5</td>
<td>59.1</td>
<td>68.1</td>
</tr>
<tr>
<td>Mask R-CNN, keypoint-only</td>
<td>62.7</td>
<td>87.0</td>
<td>68.4</td>
<td>57.4</td>
<td>71.1</td>
</tr>
<tr>
<td>Mask R-CNN, keypoint &amp; mask</td>
<td><strong>63.1</strong></td>
<td><strong>87.3</strong></td>
<td><strong>68.7</strong></td>
<td><strong>57.8</strong></td>
<td><strong>71.4</strong></td>
</tr>
</tbody>
</table>

**Table 5.** Multi-task learning of box, mask, and keypoint about the person category, evaluated on minival. All entries are trained on the same data for fair comparisons. The backbone is ResNet-50-FPN. The entries with 64.2 and 64.7 AP on minival have test-dev AP of 62.7 and 63.1, respectively (see Table 4).

<table>
<thead>
<tr>
<th></th>
<th>$\text{AP}_{bb, \text{person}}$</th>
<th>$\text{AP}_{mask, \text{person}}$</th>
<th>$\text{AP}_{kp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN</td>
<td>52.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mask R-CNN, mask-only</td>
<td>53.6</td>
<td>45.8</td>
<td>-</td>
</tr>
<tr>
<td>Mask R-CNN, keypoint-only</td>
<td>50.7</td>
<td>-</td>
<td>64.2</td>
</tr>
<tr>
<td>Mask R-CNN, keypoint &amp; mask</td>
<td>52.0</td>
<td>45.1</td>
<td><strong>64.7</strong></td>
</tr>
</tbody>
</table>

**Table 6.** RoIAlign vs. RoIPool for keypoint detection on minival. The backbone is ResNet-50-FPN.

<table>
<thead>
<tr>
<th></th>
<th>$\text{AP}_{kp}$</th>
<th>$\text{AP}_{kp, 50}$</th>
<th>$\text{AP}_{kp, 75}$</th>
<th>$\text{AP}_{kp, M}$</th>
<th>$\text{AP}_{kp, L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoIPool</td>
<td>59.8</td>
<td>86.2</td>
<td>66.7</td>
<td>55.1</td>
<td>67.4</td>
</tr>
<tr>
<td>RoIAlign</td>
<td><strong>64.2</strong></td>
<td><strong>86.6</strong></td>
<td><strong>69.7</strong></td>
<td><strong>58.7</strong></td>
<td><strong>73.0</strong></td>
</tr>
</tbody>
</table>
Table 7. Results on Cityscapes \textit{val} (‘AP [val]’ column) and \textit{test} (remaining columns) sets. Our method uses ResNet-50-FPN.

### Appendix A: Experiments on Cityscapes

We further report instance segmentation results on the Cityscapes \cite{Cordts2016Cityscapes} dataset. This dataset has fine annotations for 2975 train, 500 val, and 1525 test images. It has 20k coarse training images without instance annotations, which we do not use. All images are 2048×1024 pixels. The instance segmentation task involves 8 object categories, whose numbers of instances on the fine training set are:

<table>
<thead>
<tr>
<th>person</th>
<th>rider</th>
<th>car</th>
<th>truck</th>
<th>bus</th>
<th>train</th>
<th>mcycle</th>
<th>bicycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.9k</td>
<td>1.8k</td>
<td>26.9k</td>
<td>0.5k</td>
<td>0.4k</td>
<td>0.2k</td>
<td>0.7k</td>
<td>3.7k</td>
</tr>
</tbody>
</table>

Instance segmentation performance on this task is measured by the COCO-style mask AP (averaged over IoU thresholds); \text{AP}_{50} (i.e., mask AP at an IoU of 0.5) is also reported.

#### Implementation:
We apply our Mask R-CNN models with the ResNet-FPN-50 backbone; we found the 101-layer counterpart performs similarly due to the small dataset size. We train with image scale (shorter side) randomly sampled from [800, 1024], which reduces overfitting; inference is on a single scale of 1024 pixels. We use a mini-batch size of 1 image per GPU (so 8 on 8 GPUs) and train the model for 24k iterations, starting from a learning rate of 0.01 and reducing it to 0.001 at 18k iterations. It takes ~4 hours of training on a single 8-GPU machine under this setting.

#### Results:
Table 7 compares our results to the state of the art on the \textit{val} and \textit{test} sets. \textit{Without} using the coarse training set, our method achieves 26.2 AP on \textit{test}, which is over 30% relative improvement over the previous best entry (DIN \cite{DIN}), and is also better than the concurrent work of SGN’s 25.0 \cite{SGN}. Both DIN and SGN use \textit{fine + coarse} data. Compared to the best entry using \textit{fine data only} (17.4 AP), we achieve a \textasciitilde 50% improvement.

For the \textit{person} and \textit{car} categories, the Cityscapes dataset exhibits a large number of \textit{within}-category overlapping instances (on average 6 people and 9 cars per image). We argue that \textit{within-category overlap} is a core difficulty of instance segmentation. Our method shows massive improvement on these two categories over the other best entries (relative \textasciitilde 40% improvement on \textit{person} from 21.8 to 30.5 and \textasciitilde 20% improvement on \textit{car} from 39.4 to 46.9), even though our method does not exploit the \textit{coarse} data.

A main challenge of the Cityscapes dataset is training models in a \textit{low-data} regime, particularly for the categories of\textit{ truck, bus, and train}, which have about 200-500 train-
Table 8. Enhanced detection results of Mask R-CNN on COCO minival. Each row adds an extra component to the above row. We denote ResNeXt model by ‘X’ for notational brevity.

<table>
<thead>
<tr>
<th>description</th>
<th>backbone</th>
<th>AP</th>
<th>AP50</th>
<th>AP75</th>
<th>AP50p</th>
<th>AP75p</th>
</tr>
</thead>
<tbody>
<tr>
<td>original baseline</td>
<td>X-101-FPN</td>
<td>37.0</td>
<td>59.7</td>
<td>90.0</td>
<td>40.5</td>
<td>63.0</td>
</tr>
<tr>
<td>+ e2e training</td>
<td>X-101-FPN</td>
<td>37.6</td>
<td>60.4</td>
<td>91.7</td>
<td>41.5</td>
<td>64.4</td>
</tr>
<tr>
<td>+ ImageNet-5k</td>
<td>X-101-FPN</td>
<td>38.6</td>
<td>61.7</td>
<td>92.7</td>
<td>42.7</td>
<td>65.1</td>
</tr>
<tr>
<td>+ train-time augm.</td>
<td>X-101-FPN</td>
<td>39.2</td>
<td>62.5</td>
<td>94.6</td>
<td>43.9</td>
<td>65.9</td>
</tr>
<tr>
<td>+ deeper</td>
<td>X-152-FPN</td>
<td>39.7</td>
<td>63.2</td>
<td>95.1</td>
<td>44.1</td>
<td>66.4</td>
</tr>
<tr>
<td>+ Non-local [43]</td>
<td>X-152-FPN-NL</td>
<td>40.3</td>
<td>64.4</td>
<td>95.7</td>
<td>45.0</td>
<td>67.8</td>
</tr>
<tr>
<td>+ test-time augm.</td>
<td>X-152-FPN-NL</td>
<td>41.8</td>
<td>66.0</td>
<td>97.3</td>
<td>47.0</td>
<td>69.3</td>
</tr>
</tbody>
</table>

Table 9. Enhanced keypoint results of Mask R-CNN on COCO minival. Each row adds an extra component to the above row. Here we use only keypoint annotations but no mask annotations. We denote ResNet by ‘R’ and ResNeXt by ‘X’ for brevity.

<table>
<thead>
<tr>
<th>description</th>
<th>backbone</th>
<th>AP50</th>
<th>AP75p</th>
<th>AP50p</th>
<th>AP75p</th>
</tr>
</thead>
<tbody>
<tr>
<td>original baseline</td>
<td>R-50-FPN</td>
<td>64.2</td>
<td>86.5</td>
<td>69.7</td>
<td>58.7</td>
</tr>
<tr>
<td>+ e2e training</td>
<td>R-50-FPN</td>
<td>65.1</td>
<td>86.6</td>
<td>70.9</td>
<td>59.9</td>
</tr>
<tr>
<td>+ deeper</td>
<td>R-101-FPN</td>
<td>66.1</td>
<td>87.7</td>
<td>71.7</td>
<td>60.5</td>
</tr>
<tr>
<td>+ ResNeXt</td>
<td>X-101-FPN</td>
<td>67.3</td>
<td>88.0</td>
<td>73.3</td>
<td>62.2</td>
</tr>
<tr>
<td>+ data distillation [35]</td>
<td>X-101-FPN</td>
<td>69.1</td>
<td>89.9</td>
<td>75.3</td>
<td>64.1</td>
</tr>
<tr>
<td>+ test-time augm.</td>
<td>X-101-FPN</td>
<td>70.4</td>
<td>89.3</td>
<td>76.8</td>
<td>65.8</td>
</tr>
</tbody>
</table>

Appendix B: Enhanced Results on COCO

As a general framework, Mask R-CNN is compatible with complementary techniques developed for detection/segmentation, including improvements made to Fast/Faster R-CNN and FCNs. In this appendix we describe some techniques that improve over our original results. Thanks to its generality and flexibility, Mask R-CNN was used as the framework by the three winning teams in the COCO 2017 instance segmentation competition, which all significantly outperformed the previous state of the art.

Instance Segmentation and Object Detection

We report some enhanced results of Mask R-CNN in Table 8. Overall, the improvements increase mask AP 5.1 points (from 36.7 to 41.8) and box AP 7.7 points (from 39.6 to 47.3). Each model improvement increases both mask AP and box AP consistently, showing good generalization of the Mask R-CNN framework. We detail the improvements next. These results, along with future updates, can be reproduced by our released code at https://github.com/facebookresearch/Detectron, and can serve as higher baselines for future research.

Updated baseline: We start with an updated baseline with a different set of hyper-parameters. We lengthen the training to 180k iterations, in which the learning rate is reduced by 10 at 120k and 160k iterations. We also change the NMS threshold to 0.5 (from a default value of 0.3). The updated baseline has 37.0 mask AP and 40.5 box AP.

End-to-end training: All previous results used stage-wise training, i.e., training RPN as the first stage and Mask R-CNN as the second. Following [37], we evaluate end-to-end (‘e2e’) training that jointly trains RPN and Mask R-CNN. We adopt the ‘approximate’ version in [37] that only computes partial gradients in the RoIAlign layer by ignoring the gradient w.r.t. RoI coordinates. Table 8 shows that e2e training improves mask AP by 0.6 and box AP by 1.2.

ImageNet-5k pre-training: Following [45], we experiment with models pre-trained on a 5k-class subset of ImageNet (in contrast to the standard 1k-class subset). This 5x increase in pre-training data improves both mask and box 1 AP. As a reference, [40] used ~250x more images (300M) and reported a 2-3 box AP improvement on their baselines.

Train-time augmentation: Scale augmentation at train time further improves results. During training, we randomly sample a scale from [640, 800] pixels and we increase the number of iterations to 260k (with the learning rate reduced by 10 at 200k and 240k iterations). Train-time augmentation improves mask AP by 0.6 and box AP by 0.8.

Model architecture: By upgrading the 101-layer ResNetXt to its 152-layer counterpart [19], we observe an increase of 0.5 mask AP and 0.6 box AP. This shows a deeper model can still improve results on COCO.

Using the recently proposed non-local (NL) model [43], we achieve 40.3 mask AP and 45.0 box AP. This result is without test-time augmentation, and the method runs at 3fps on an Nvidia Tesla P100 GPU at test time.

Test-time augmentation: We combine the model results evaluated using scales of [400, 1200] pixels with a step of 100 and on their horizontal flips. This gives us a single-model result of 41.8 mask AP and 47.3 box AP.

The above result is the foundation of our submission to the COCO 2017 competition (which also used an ensemble, not discussed here). The first three winning teams for the instance segmentation task were all reportedly based on an extension of the Mask R-CNN framework.

Keypoint Detection

We report enhanced results of keypoint detection in Table 9. As an updated baseline, we extend the training schedule to 130k iterations in which the learning rate is reduced by 10 at 100k and 120k iterations. This improves AP50p by about 1 point. Replacing ResNet-50 with ResNet-101 and ResNeXt-101 increases AP50p to 66.1 and 67.3, respectively.

With a recent method called data distillation [35], we are able to exploit the additional 120k unlabeled images provided by COCO. In brief, data distillation is a self-training strategy that uses a model trained on labeled data to predict annotations on unlabeled images, and in turn updates the model with these new annotations. Mask R-CNN provides an effective framework for such a self-training strategy. With data distillation, Mask R-CNN AP50p improve by 1.8 points to 69.1. We observe that Mask R-CNN can benefit from extra data, even if that data is unlabeled.

By using the same test-time augmentation as used for instance segmentation, we further boost AP50p to 70.4.
Acknowledgements: We would like to acknowledge Ilija Radosavovic for contributions to code release and enhanced results, and the Caffe2 team for engineering support.

References

[38] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In CVPR, 2016. 2, 5


