

Mask R-CNN

Kaiming He^{ID}, Georgia Gkioxari, Piotr Dollár, and Ross Girshick

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>.

Index Terms—Instance segmentation, object detection, pose estimation, convolutional neural network

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 [1], [2] and Fully Convolutional Network (FCN) [3] 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.¹ 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 [2] 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 (Fig. 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* [1], [5], 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 a seemingly minor change, *RoIAlign* has a large impact: it improves mask accuracy by relative 10 to 50 percent, 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 [6], 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.

1. Following 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.

• The authors are with Facebook AI Research, Menlo Park, CA 94025-1452 USA. E-mail: {kaiminghe, gkioxari, pdollar, rbg}@fb.com.

Manuscript received 15 Jan. 2018; revised 7 May 2018; accepted 9 May 2018. Date of publication 5 June 2018; date of current version 6 Jan. 2020.

(Corresponding author: Kaiming He.)

Recommended for acceptance by R. Cucchiara, N. Sebe, S. Soatto, and Y. Matsushita.

Digital Object Identifier no. 10.1109/TPAMI.2018.2844175

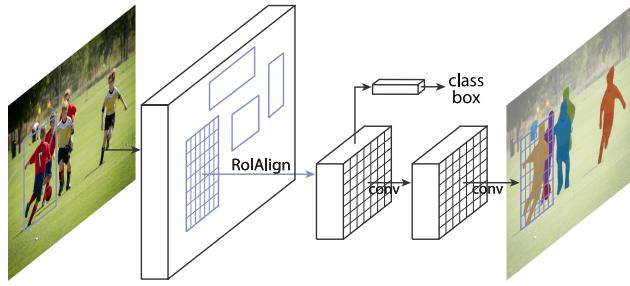


Fig. 1. The *Mask R-CNN* framework for instance segmentation.

Our models can run at about 200 ms 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 [6]. 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.

A preliminary version of this manuscript was published previously [7]. As a generic framework, Mask R-CNN is compatible with complementary techniques developed for detection/segmentation, as have been widely witnessed in Fast/Faster R-CNN and FCN in the past years. This manuscript also describes some techniques that improve over our original results published in [7]. 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.

We have released code to facilitate future research.

2 RELATED WORK

R-CNN. The Region-based CNN (R-CNN) approach [8] to bounding-box object detection is to attend to a manageable number of candidate object regions [9], [10] and evaluate convolutional networks [11], [12] independently on each ROI. R-CNN was extended [1], [5] to allow attending to ROIs on feature maps using RoIPool, leading to fast speed and better accuracy. Faster R-CNN [2] 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., [13], [14], [15]), 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 [8], [16], [17], [18] resorted to bottom-up segments [9], [19]. DeepMask [20] and following works [21], [22] 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.[23] 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.[24] combined the segment proposal system in [22] and object detection system in [25] for “fully convolutional instance segmentation” (FCIS). The common idea in [22], [24], [25] 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 (Fig. 6), showing that it is challenged by the fundamental difficulties of segmenting instances.

Another family of solutions [26], [27], [28], [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 [2]. Faster R-CNN consists of two stages. The first stage, called a Region Proposal Network, proposes candidate object bounding boxes. The second stage, which is in essence Fast R-CNN [1], 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 [15] 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.[20], [23], [24]). Our approach follows the spirit of Fast R-CNN [1] that applies bounding-box classification and regression in *parallel* (which turned out to largely simplify the multi-stage pipeline of original R-CNN [8]).

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 [1]. The mask branch has a Km^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

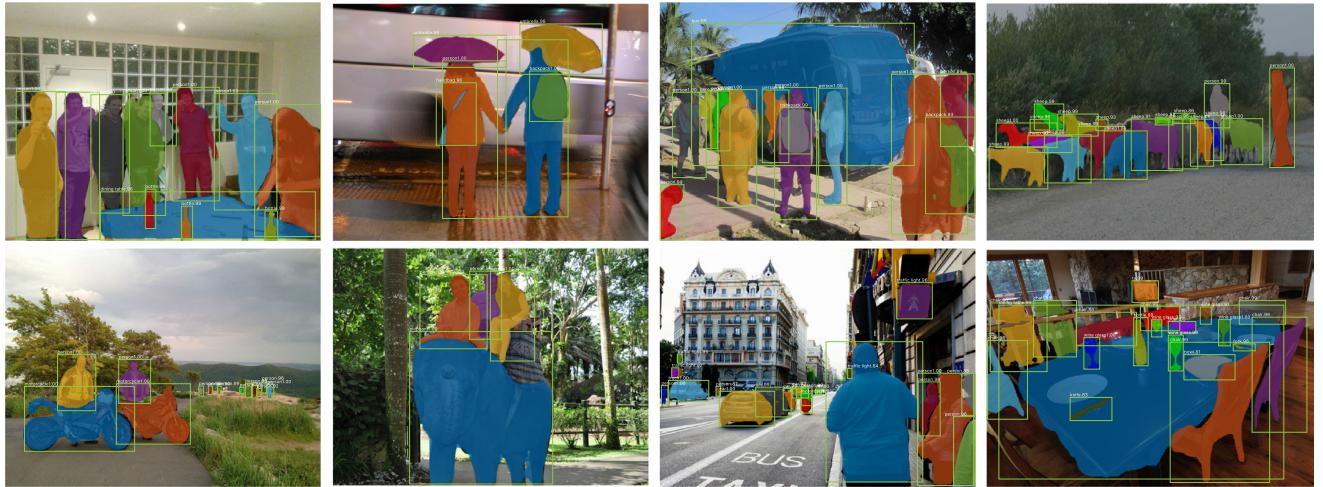


Fig. 2. *Mask R-CNN* results on the COCO test set. These results are based on ResNet-101 [4], achieving a *mask AP* of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

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 [3] 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 [3]. 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 [20], [21], [23], 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 [1] is a standard operation for extracting a small feature map (e.g., 7×7) 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 $[x/16]$, where 16 is a feature map stride and $[.]$ is rounding; likewise, quantization is

performed when dividing into bins (e.g., 7×7). 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 bilinear interpolation [31] 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).² See Fig. 3 for our implementation details. We note that the results are not sensitive to where the four sampling points are located in the bin, or how many points are sampled, *as long as* no quantization is performed on any coordinates involved.

RoIAlign leads to large improvements as we show in Section 4.2. We also compare to the *RoIWarp* operation proposed in [23]. Unlike *RoIAlign*, *RoIWarp* overlooked the alignment issue and was implemented in [23] as quantizing RoI just like *RoIPool*. So even though *RoIWarp* also adopts bilinear resampling motivated by [31], it performs on par with *RoIPool* as shown by experiments (more details in Table 2 c), 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 [4] and ResNeXt [32] networks of depth 50 or 101 layers. The original implementation of Faster R-CNN with ResNets [4] extracted features from the final convolutional layer of the

2. We sample four regular locations, so that we can evaluate either max or average pooling. In fact, interpolating only a single value at each bin center (without pooling) is nearly as effective.

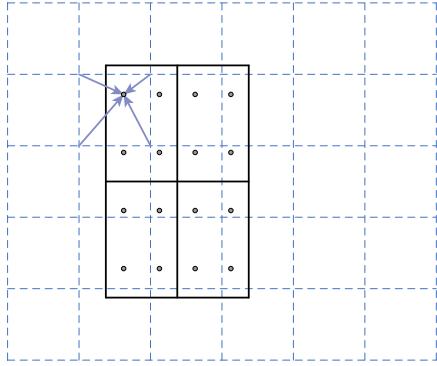


Fig. 3. Implementation of *RoIAlign*. The dashed grid is a feature map on which *RoIAlign* is performed, the solid lines illustrate an ROI (with 2×2 bins in this example), and the dots denote the 4 sampling points within each bin. The value of each sampling point is computed by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the ROI, its bins, or the sampling points.

4th 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 [4], [15], [23], [33].

We also explore another more effective backbone recently proposed by Lin et al. [14], 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 [14].

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 [4] and FPN [14] papers. Details are shown in Fig. 4. The head on the ResNet-C4 backbone includes the 5th stage of ResNet (namely, the 9-layer ‘res5’ [4]), 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.

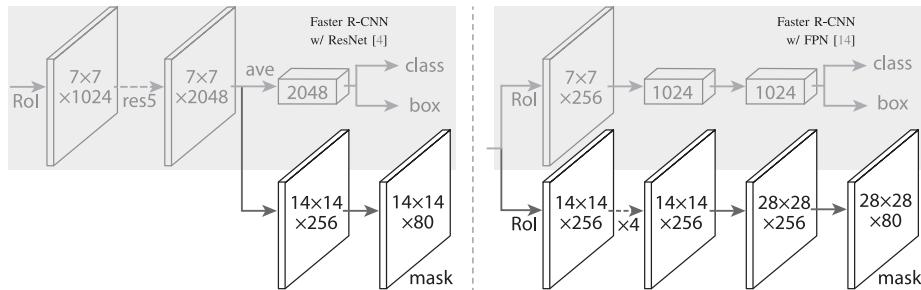


Fig. 4. *Head Architecture*: We extend two existing Faster R-CNN heads [4], [14]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [4] and [14], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, 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 , deconv 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 [4]). *Right: ‘ $\times 4$* denotes a stack of four consecutive convs.

3.1 Implementation Details

We set hyper-parameters following existing Fast/Faster R-CNN work [1], [2], [14]. Although these decisions were made for object detection in original papers [1], [2], [14], 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 [1]. Images are resized such that their scale (shorter edge) is 800 pixels [14]. 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 [1]. N is 64 for the C4 backbone (as in [1], [2]) and 512 for FPN (as in [14]). 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 a momentum of 0.9. When using ResNeXt [32], we use a mini-batch size of 1 image per GPU with the same number of iterations, and a starting learning rate of 0.01.

The RPN anchors span 5 scales and 3 aspect ratios, following [14]. 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 [2]) and 1,000 for FPN (as in [14]). We run the box prediction branch on these proposals, followed by non-maximum suppression [34]. 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., \raise.17ex\~20% on typical models).

TABLE 1
Instance Segmentation Mask AP on COCO test-dev

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
MNC [23]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [24] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [24] +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

MNC [23] and FCIS [24] 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 [13]. All entries are single-model results.



Fig. 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).

4 EXPERIMENTS: INSTANCE SEGMENTATION

We perform a thorough comparison of Mask R-CNN to the state of the art along with comprehensive ablation experiments. We use the COCO dataset [6] for all experiments. We report the standard COCO metrics including AP (averaged over IoU thresholds), AP₅₀, AP₇₅, and AP_S, AP_M, AP_L (AP at different scales). Unless otherwise noted, AP is evaluating using *mask* IoU. As in previous work [14], [35], we train using the union of 80k train images and a 35k subset of val images (trainval35k), and report ablations on the remaining 5k subset of val images (minival). We also report results on test-dev [6], which has no disclosed labels.

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 [23] and FCIS [24], the winners of the COCO 2015 and 2016 segmentation challenges, respectively. Without bells and whistles, Mask R-CNN with ResNet-101-FPN backbone outperforms FCIS++ [24], which includes multi-scale train/test, horizontal flip

test, and online hard example mining (OHEM) [13]. While outside the scope of this work, we expect many such improvements to be applicable to ours.

Mask R-CNN outputs are visualized in Figs. 2 and 5. Mask R-CNN achieves good results even under challenging conditions. In Fig. 6 we compare our Mask R-CNN baseline and FCIS+++ [24]. FCIS+++ exhibits systematic artifacts on overlapping instances, suggesting that it is challenged by the fundamental difficulty of instance segmentation. Mask R-CNN shows no such artifacts.

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 2 a shows Mask R-CNN with various backbones. It benefits from deeper networks (50 versus 101) and advanced designs including FPN and ResNeXt.³ We note that not all frameworks automatically benefit from deeper or advanced networks (see benchmarking in [15]).

3. We use the 64×4d variant of ResNeXt [32].

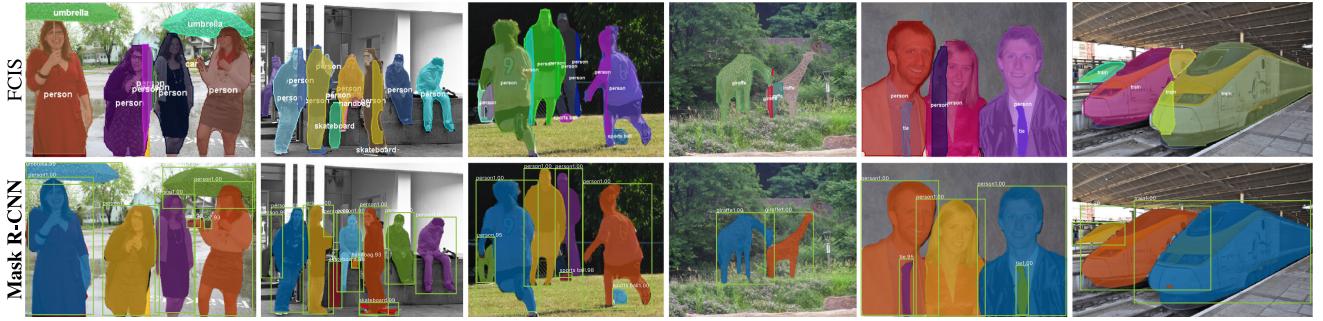


Fig. 6. FCIS+++ [24] (top) versus Mask R-CNN (bottom, ResNet-101-FPN). FCIS exhibits systematic artifacts on overlapping objects.

TABLE 2
Ablations

<i>net-depth-features</i>	AP	AP ₅₀	AP ₇₅
ResNet-50-C4	30.3	51.2	31.5
ResNet-101-C4	32.7	54.2	34.3
ResNet-50-FPN	33.6	55.2	35.3
ResNet-101-FPN	35.4	57.3	37.5
ResNeXt-101-FPN	36.7	59.5	38.9

(a) **Backbone Architecture:** Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.

	AP	AP ₅₀	AP ₇₅
<i>softmax</i>	24.8	44.1	25.1
<i>sigmoid</i>	30.3	51.2	31.5

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

	AP	AP ₅₀	AP ₇₅	AP ^{bb}	AP ₅₀ ^{bb}	AP ₇₅ ^{bb}
<i>RoIPool</i>	23.6	46.5	21.6	28.2	52.7	26.9
<i>RoIAlign</i>	30.9	51.8	32.1	34.0	55.3	36.4
	+7.3	+5.3	+10.5	+5.8	+2.6	+9.5

(d) **RoIAlign** (ResNet-50-C5, stride 32): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in big accuracy gaps.

	<th>bilinear? </th> <th>agg. </th> <th>AP</th> <th>AP₅₀</th> <th>AP₇₅</th>	bilinear?	agg.	AP	AP ₅₀	AP ₇₅
<i>RoIPool</i> [1]			max	26.9	48.8	26.4
<i>RoIWarp</i> [23]		✓	max	27.2	49.2	27.1
		✓	ave	27.1	48.9	27.1
<i>RoIAlign</i>	✓	✓	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 AP₇₅ by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

	mask branch		AP	AP ₅₀	AP ₇₅
MLP	fc:	1024→1024→80·28 ²	31.5	53.7	32.8
MLP	fc:	1024→1024→1024→80·28 ²	31.5	54.0	32.6
FCN	conv:	256→256→256→256→256→80	33.6	55.2	35.3

(e) **Mask Branch** (ResNet-50-FPN): Fully convolutional networks (FCN) *vs.* multi-layer perceptrons (MLP, fully-connected) for mask prediction. FCNs improve results as they take advantage of explicitly encoding spatial layout.

We train on *trainval35k*, test on *minival*, and report mask AP unless otherwise noted.

Multinomial versus 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 (by a per-pixel *sigmoid* and a *binary loss*). In Table 2 b, we compare this to using a per-pixel *softmax* and a *multinomial loss* (as commonly used in FCN [3]). This alternative *couples* the tasks of mask and class prediction, and results in a severe loss in mask AP (5.5 points). This suggests that once the instance has been classified as a whole (by the box branch), it is sufficient to predict a binary mask without concern for the categories, which makes the model easier to train.

Class-Specific versus 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 versus 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 2 c. 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 (AP₇₅). *RoIAlign* is insensitive to max/average pool; we use average in the rest of the paper.

Additionally, we compare with *RoIWarp* proposed in MNC [23] that also adopt bilinear sampling. As discussed in Section 3, *RoIWarp* still quantizes the RoI, losing alignment with the input. As can be seen in Table 2 c, *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 Fig. 4 (right), as the res5 head is not applicable. Table 2 d shows that *RoIAlign* improves mask AP by a massive 7.3 points, and mask AP₇₅ by 10.5 points (*50 percent 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 2 c). *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 7).

Mask Branch. Segmentation is a pixel-to-pixel task and we exploit the spatial layout of masks by using an FCN. In Table 2 e, 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.

TABLE 3
Object Detection Single-Model Results (Bounding Box AP), versus State-of-the-Art on test-dev

	backbone	AP ^{bb}	AP ₅₀ ^{bb}	AP ₇₅ ^{bb}	AP _S ^{bb}	AP _M ^{bb}	AP _L ^{bb}
Faster R-CNN+++ [4]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [14]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [15]	Inception-ResNet-v2 [36]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [33]	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

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 [14] come from using RoIAlign (+1.1 AP^{bb}), multitask training (+0.9 AP^{bb}), and ResNeXt-101 (+1.6 AP^{bb}).

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 [15], 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 [33] (which used Inception-ResNet-v2-TDM).

As a further comparison, we trained a version of Mask R-CNN but *without* the mask branch, denoted by “Faster R-CNN, RoIAlign” in Table 3. This model performs better than the model presented in [14] 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 [2]. This model runs at 195 ms per image on an Nvidia Tesla M40 GPU (plus 15 ms

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 ~400 ms as it has a heavier box head (Fig. 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 [15], 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.

4.5 Advanced Results

Mask R-CNN is a generic framework and is compatible with orthogonal improvements that have been seen in many other detection/segmentation systems. For completeness we report some advanced results of Mask R-CNN in Table 4. Results in this table can be reproduced by our released code (<https://github.com/facebookresearch/Detectron>), which can serve as higher baselines for future research.

Overall, the improvements we implemented achieve in total 5.1 points mask AP (from 36.7 to 41.8) and 7.7 points box AP (from 39.6 to 47.3) increase. Each component of the improvements in general increases both mask AP and box

TABLE 4
Advanced Results of Mask R-CNN on COCO minival, Showing Both Mask AP and Bounding Box AP

description	backbone	AP	AP ₅₀	AP ₇₅	AP ^{bb}	AP ₅₀ ^{bb}	AP ₇₅ ^{bb}
original baseline	ResNeXt-101-FPN	36.7	59.5	38.9	39.6	61.5	43.2
+ updated baseline	ResNeXt-101-FPN	37.0	59.7	39.0	40.5	63.0	43.7
+ e2e training	ResNeXt-101-FPN	37.6	60.4	39.9	41.7	64.1	45.2
+ ImageNet-5k	ResNeXt-101-FPN	38.6	61.7	40.9	42.7	65.1	46.6
+ train-time augmentation	ResNeXt-101-FPN	39.2	62.5	41.6	43.5	65.9	47.2
+ deeper	ResNeXt-152-FPN	39.7	63.2	42.2	44.1	66.4	48.4
+ Non-local [37]	ResNeXt-152-FPN-NL	40.3	64.4	42.8	45.0	67.8	48.9
+ test-time augmentation	ResNeXt-152-FPN-NL	41.8	66.0	44.8	47.3	69.3	51.5

Each line shows an extra component incrementally made to the line above.

AP consistently, showing good generalization of Mask R-CNN as a framework.

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. The details of other improvements in Table 4 are as follows.

End-to-End Training. The results presented so far were based on step-wise training, i.e., training RPN as the first step and Mask R-CNN as the second. Following [38], we also experiment with end-to-end (“e2e”) training that jointly trains RPN and Mask R-CNN. We adopt the “approximate” version in [38] that only computes partial gradients in the RoIAlign layer by ignoring the gradient w.r.t.RoI coordinates. Table 4 shows that e2e training improves mask AP by 0.6 points and box AP by 1.2 points.

ImageNet-5k Pre-Training. In the following, we experiment with models pre-trained on a 5k-class subset of ImageNet, following [32], in contrast to the standard 1k-class subset. This improves both mask and box AP by 1 point, contributed by $\sim 5 \times$ more pre-training data. As a reference, a recent work [39] using $\sim 250 \times$ more images (300M) reported 2-3 points improvement of box AP on their Faster R-CNN baselines.

Train-Time Augmentation. Next we improve the results by using scale augmentation at training time, noting that the results so far were trained on a single scale of 800 pixels. During training, we randomly sample a scale from [640, 800] pixels. For the model to be sufficiently trained, 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 points and box AP by 0.8 points.

Model Architectures. By upgrading the 101-layer ResNeXt to its 152-layer counterpart [4], we have seen an increase of 0.5 points for mask AP and 0.6 points for box AP. Despite showing a small gap, a deeper model can still improve both segmentation and detection on the COCO dataset.

Using the recently proposed *non-local* (NL) model [37], we have achieved 40.3 mask AP and 45.0 box AP. This result is without test-time augmentation, and the method runs at 3 fps in an Nvidia Tesla P100 GPU at test time.

Test-Time Augmentation. Finally, we evaluate test-time augmentation results on the above entry. We combine the results evaluated on images with scales of [400, 1200] pixels

at a step of 100 pixels and on their horizontal flipping. 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. The first 3 winning teams for the instance segmentation task were all reportedly based on an extension of the Mask R-CNN framework.

In summary, the Mask R-CNN framework can easily enjoy orthogonal improvements that have been witnessed on many detection/segmentation systems, including end-to-end training, larger-scale data, train/test-time augmentation, and better backbone architectures. We expect that future advances on these orthogonal directions will benefit the Mask R-CNN framework further.

5 EXPERIMENTS: KEYPOINT 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 [40]) will be complementary to our simple approach, but it is beyond the scope of this paper.

5.1 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 single point to be detected). We note that as in instance segmentation, the K keypoints are still treated independently.

We adopt the ResNet-FPN variant, and the keypoint head architecture is similar to that in Fig. 4 (right). The keypoint head consists of a stack of eight 3×3 512-d conv layers, followed by a deconv layer and $2 \times$ bilinear upscaling, producing an output resolution of 56×56 . We found that a relatively high resolution output (compared to masks) is required for keypoint-level localization accuracy.

TABLE 5
Keypoint Detection AP on COCO test-dev

	backbone	AP ^{kp}	AP ^{kp} ₅₀	AP ^{kp} ₇₅	AP ^{kp} _M	AP ^{kp} _L
CMU-Pose+++ [40]	VGG	61.8	84.9	67.5	57.1	68.2
G-RMI [41] [†]	Inception-ResNet-v2 + ResNet-101	62.4	84.0	68.5	59.1	68.1
Mask R-CNN , keypoint-only	ResNet-50-FPN	62.7	87.0	68.4	57.4	71.1
Mask R-CNN , keypoint & mask	ResNet-50-FPN	63.1	87.3	68.7	57.8	71.4

Ours (ResNet-50-FPN) is a single model that runs at 5 fps. CMU-Pose+++ [40] is the 2016 competition winner that uses multi-scale testing, post-processing with CPM [42], and filtering with an object detector, adding a cumulative ~ 5 points (clarified in personal communication). [†]: G-RMI was trained on COCO plus MPII [43] (25k images), using two models (Inception-ResNet-v2 for bounding box detection and ResNet-101 for keypoints).

TABLE 6

Multi-Task Learning of Box, Mask, and Keypoint About the Person Category, Evaluated on minival

	AP ^{bb} person	AP ^{mask} person	AP ^{kp}
Faster R-CNN	52.5	-	-
Mask R-CNN, mask-only	53.6	45.8	-
Mask R-CNN, keypoint-only	50.7	-	64.2
Mask R-CNN, keypoint & mask	52.0	45.1	64.7

All entries are trained on the same data for fair comparisons. The backbone is ResNet-50-FPN. The entry with 64.2 AP on minival has 62.7 AP on test-dev. The entry with 64.7 AP on minival has 63.1 AP on test-dev (see Table 5).

Models are trained on all COCO trainval35k images that contain annotated keypoints. To reduce overfitting, as this training set is smaller, we train the models 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 non-maximum suppression with a threshold of 0.5. Other implementations are identical as in Section 3.1.

5.2 Main Results and Ablations

We evaluate the person keypoint AP (AP^{kp}) and experiment with a ResNet-50-FPN backbone; more backbones will be studied in the next section. Table 5 shows that our result (62.7 AP^{kp}) is 0.9 points higher than the COCO 2016 keypoint detection winner [40] that uses a multi-stage processing pipeline (see caption of Table 5). 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 AP^{kp} to 63.1 (Table 5) on test-dev. More ablations of multi-task learning on minival are in Table 6. 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 (Fig. 7).

We also investigate the effect of *RoIAlign* on keypoint detection (Table 7). Though this ResNet-50-FPN backbone

TABLE 7

RoIPool versus RoIAlign for Keypoint Detection on minival

	AP ^{kp}	AP ^{kp} ₅₀	AP ^{kp} ₇₅	AP ^{kp} _M	AP ^{kp} _L
RoIPool	59.8	86.2	66.7	55.1	67.4
RoIAlign	64.2	86.6	69.7	58.7	73.0

The backbone is ResNet-50-FPN.

has finer strides (e.g., 4 pixels on the finest level), RoIAlign still shows significant improvement over RoIPool and increases 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.

5.3 Advanced Results

In addition to the baseline results, we report advanced results of keypoint detection in Table 8.

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 AP^{kp} by about 1 point. On top of this baseline, replacing ResNet-50 with ResNet-101 and ResNeXt-101 respective increases AP^{kp} to 66.1 and 67.3.

With a recent method called *data distillation* [44], we are able to exploit the additional 120k *unlabeled* images officially provided by the COCO dataset. 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 by these new annotations. Mask R-CNN provides an easy-to-use framework for such a self-training strategy. With data distillation, we improve the Mask R-CNN result by 1.8 points AP^{kp} to 69.1. This result demonstrates that Mask R-CNN manages to enjoy benefit from more data, even if the extra data can be *unlabeled*.

Finally, by using the same test-time augmentation as used for instance segmentation, we further boost AP^{kp} to 70.4.

6 MORE EXPERIMENTS ON CITYSCAPES

We further report instance segmentation results on the Cityscapes [46] dataset. This dataset has fine annotations for



Fig. 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 8
Advanced Results of Mask R-CNN for Keypoint Detection on minival

description	backbone	AP ^{kp}	AP ^{kp} ₅₀	AP ^{kp} ₇₅	AP ^{kp} _M	AP ^{kp} _L
original baseline	ResNet-50-FPN	64.2	86.6	69.7	58.7	73.0
+ updated baseline	ResNet-50-FPN	65.1	86.6	70.9	59.9	73.6
+ deeper	ResNet-101-FPN	66.1	87.7	71.7	60.5	75.0
+ ResNeXt	ResNeXt-101-FPN	67.3	88.0	73.3	62.2	75.6
+ data distillation [44]	ResNeXt-101-FPN	69.1	88.9	75.3	64.1	77.1
+ test-time augmentation	ResNeXt-101-FPN	70.4	89.3	76.8	65.8	78.1

Each line shows an extra component incrementally made to the line above. Here we use only keypoint annotations but no mask annotations.

2,975 training images, 500 validation images, and 1,525 test images. It has 20k coarse training images without instance annotations, which we do *not* use. All images are at a fixed resolution of 2048 × 1024 pixels. The instance segmentation task involves 8 object categories, whose numbers of instances on the fine training set are:

Instance segmentation performance on this task is measured by the COCO-style mask AP (averaged over IoU thresholds); AP₅₀ (i.e., mask AP at an IoU of 0.5) is also reported.

6.1 Implementation

We apply our Mask R-CNN models with the ResNet-FPN-50 backbone; we have tested the 101-layer counterpart and found it 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 1,024 pixels. We use a mini-batch size of 1 image per GPU (so effectively 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. Other implementation details are identical as in Section 3.1.

6.2 Results

Table 9 compares our results to the state of the art on the val and test sets. *Without* using the coarse training set, our method achieves 26.2 AP on test, which is over 30 percent relative improvement over the previous best entry (DIN [28]), and is also better than the concurrent work of SGN’s 25.0 [29]. Both DIN and SGN use fine + coarse data. Compared to the best entry using fine data only (17.4 AP), we achieve a ~50% improvement.

person	rider	car	truck	bus	train	mcycle	bicycle
17.9k	1.8k	26.9k	0.5k	0.4k	0.2k	0.7k	3.7k

For the *person* and *car* categories, the Cityscapes dataset exhibits a large number of *within-category* overlapping instances (on average 6 people and 9 cars per image). We argue that *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 ~40% improvement on *person* from 21.8 to 30.5 and ~20% improvement on *car* from 39.4 to 46.9), even though our method does not exploit the coarse data.

A main challenge of the Cityscapes dataset is training models in a *low-data* regime, particularly for the categories of *truck*, *bus*, and *train*, which have about 200-500 training samples each. To partially remedy this issue, we further report a result using COCO pre-training. To do this, we initialize the corresponding 7 categories in Cityscapes from a pre-trained COCO Mask R-CNN model (*rider* being randomly initialized). We fine-tune this model for 4k iterations in which the learning rate is reduced at 3k iterations, which takes ~1 hour for training given the COCO model.

The COCO pre-trained Mask R-CNN model achieves 32.0 AP on test, almost a 6 point absolute improvement over the fine-only counterpart. This indicates the important role the amount of training data plays. It also suggests that instance segmentation methods on Cityscapes might be influenced by their *low-shot* learning performance. We show that using COCO pre-training is an effective strategy for mitigating the limited data issue involving this dataset.

Finally, we observed a bias between the val and test AP, as is also observed from the results of [26], [27], [29]. We found that this bias is mainly caused by the *truck*, *bus*,

TABLE 9
Results on Cityscapes val ('AP [val]' Column) and test (Remaining Columns) Sets

	training data	AP [val]	AP	AP ₅₀	person	rider	car	truck	bus	train	mcycle	bicycle
InstanceCut [26]	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 [27]	fine	19.8	15.6	30.0	15.1	11.7	32.9	17.1	20.4	15.0	7.9	4.9
SAIS [45]	fine	-	17.4	36.7	14.6	12.9	35.7	16.0	23.2	19.0	10.3	7.8
DIN [28]	fine + coarse	-	20.0	38.8	16.5	16.7	25.7	20.6	30.0	23.4	17.1	10.1
SGN [29]	fine + coarse	29.2	25.0	44.9	21.8	20.1	39.4	24.8	33.2	30.8	17.7	12.4
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

Our method uses ResNet-50-FPN. All entries use ImageNet pre-training; the last row also uses COCO to pre-train the instance segmentation model.

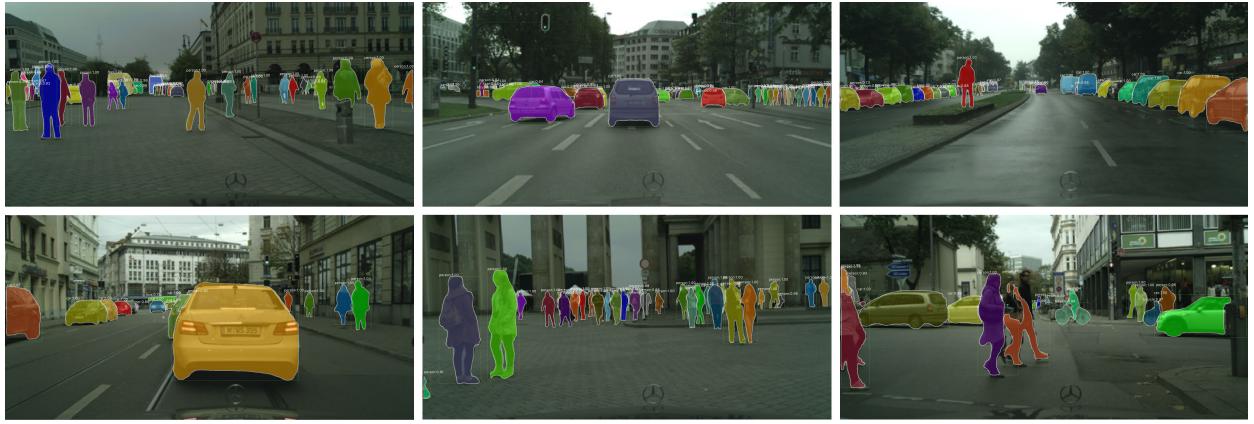


Fig. 8. Mask R-CNN results on Cityscapes test (32.0 AP). The bottom-right image shows a failure prediction.

and *train* categories, with the fine-only model having *val/test* AP of 28.8/22.8, 53.5/32.2, and 33.0/18.6, respectively. This suggests that there is a *domain shift* on these categories, which also have little training data. COCO pre-training helps to improve results the most on these categories; however, the domain shift persists with 38.0/30.1, 57.5/40.9, and 41.2/30.9 *val/test* AP, respectively. Note that for the *person* and *car* categories we do not see any such bias (*val/test* AP are within ± 1 point).

Example results on Cityscapes are shown in Fig. 8.

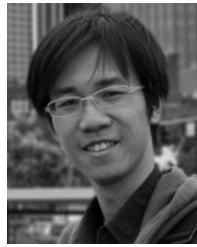
7 CONCLUSION

We have presented a simple and effective framework for instance segmentation, which also shows good results on bounding box detection and can be extended to pose estimation. We hope the simplicity and generality of this framework will facilitate future research on these and other *instance-level* visual recognition tasks.

REFERENCES

- [1] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
- [2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
- [3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
- [4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [5] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 346–361.
- [6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.
- [7] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.
- [8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.
- [9] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, pp. 154–171, 2013.
- [10] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 814–830, Apr. 2016.
- [11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
- [12] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
- [13] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 761–769.
- [14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 936–944.
- [15] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al., "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3296–3297.
- [16] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 297–312.
- [17] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 447–456.
- [18] J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3992–4000.
- [19] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 328–335.
- [20] P. O. Pinheiro, R. Collobert, and P. Dollar, "Learning to segment object candidates," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1990–1998.
- [21] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 75–91.
- [22] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 534–549.
- [23] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3150–3158.
- [24] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4438–4446.
- [25] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016.
- [26] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother, "InstanceCut: From edges to instances with MultiCut," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7322–7331.
- [27] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2858–2866.
- [28] A. Arnab and P. H. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 879–888.
- [29] S. Liu, J. Jia, S. Fidler, and R. Urtasun, "SGN: Sequential grouping networks for instance segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3516–3524.

- [30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
- [31] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015.
- [32] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5987–5995.
- [33] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta, "Beyond skip connections: Top-down modulation for object detection," *arXiv:1612.06851*, 2016.
- [34] R. Girshick, F. Iandola, T. Darrell, and J. Malik, "Deformable part models are convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 437–446.
- [35] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2874–2883.
- [36] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. Int. Conf. Learn. Representations Workshop*, 2016.
- [37] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7794–7803.
- [38] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
- [39] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 843–852.
- [40] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1302–1310.
- [41] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy, "Towards accurate multi-person pose estimation in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3711–3719.
- [42] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4724–4732.
- [43] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D human pose estimation: New benchmark and state of the art analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3686–3693.
- [44] I. Radosavovic, P. Dollár, R. Girshick, G. Gkioxari, and K. He, "Data distillation: Towards omni-supervised learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4119–4128.
- [45] Z. Hayder, X. He, and M. Salzmann, "Boundary-aware instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5696–5704.
- [46] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3213–3223.



Kaiming He received the PhD degree. He joined Facebook AI Research (FAIR) in 2016 as a research scientist. Formerly he was with Microsoft Research Asia (MSRA), which he joined in 2011. His research interests include computer vision and deep learning. He has received the Best Paper Award in CVPR 2009, CVPR 2016, and ICCV 2017.



Georgia Gkioxari received the PhD degree in computer science and electrical engineering from the University of California at Berkeley under the supervision of Jitendra Malik, in 2016. She is a research scientist with Facebook AI Research (FAIR). Her research interests lie in computer vision, with a focus on object and person recognition from static images and videos. In 2017, she received the Marr Prize at ICCV for "Mask R-CNN".



Piotr Dollár received the PhD degree under the guidance of Serge Belongie from the University of California San Diego, in 2007 and has continued doing research in vision and learning since. He is a research scientist with Facebook AI Research (FAIR) with a focus on computer vision and machine learning. Prior, he spent three years at Microsoft Research (MSR). He helped cofound Anchovi Labs (acquired by Dropbox in 2012) and before that was a postdoc with the Computation Vision Lab at Caltech until 2011.



Ross Girshick received the PhD degree in computer science from the University of Chicago under Pedro Felzenszwalb, in 2012. He is a research scientist with Facebook AI Research (FAIR), working on computer vision and machine learning. Prior, he was a researcher with Microsoft Research and a postdoc with the University of California, Berkeley, where he was advised by Jitendra Malik and Trevor Darrell. He received the 2017 PAMI Young Researcher Award and the Marr Prize at ICCV 2017 for "Mask R-CNN".

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.