2018

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.[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.

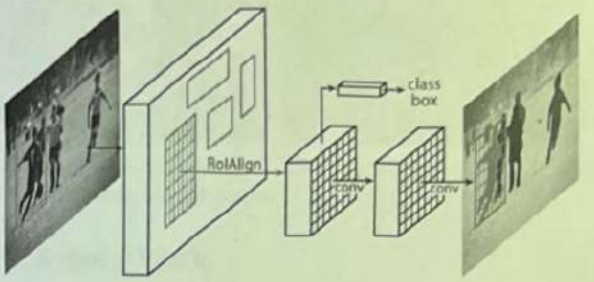

Figure 1. The **Mask R-CNN** framework for instance segmentation.

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

[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.

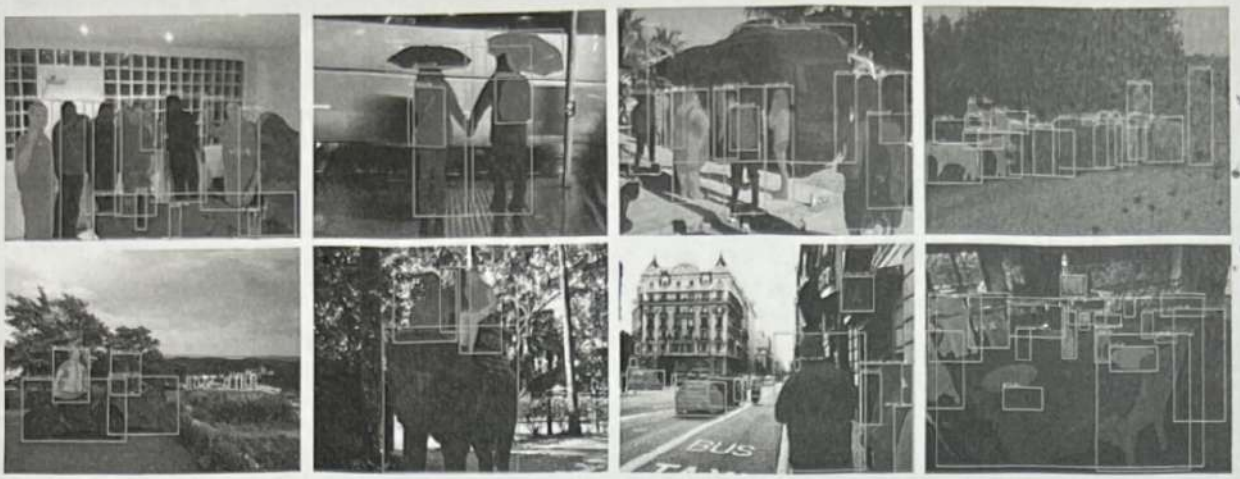

Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

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 [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 [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 [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 [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.* [33, 10, 26]). Our approach follows the spirit of Fast R-CNN [12] that applies bounding-box classification and regression in *parallel* (which turned out to largely simplify the multi-stage pipeline of original R-CNN [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 [12]. 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 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 [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.

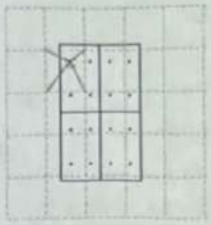

Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

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 [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 [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 [12] 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 $[\cdot]$ 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 [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 RoIWarp operation proposed in [10]. Unlike RoIAlign, RoIWarp overlooked the alignment issue and was implemented in [10] as quantizing RoI just like RoIPool. So even though RoIWarp also adopts bilinear 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.

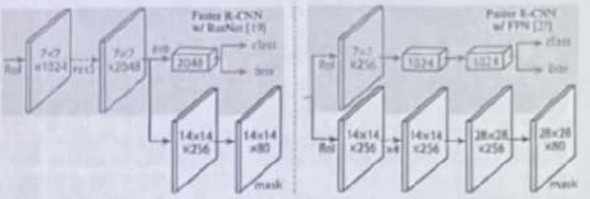

Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or *fc* layers as can be inferred from context (conv preserves spatial dimension while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.

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

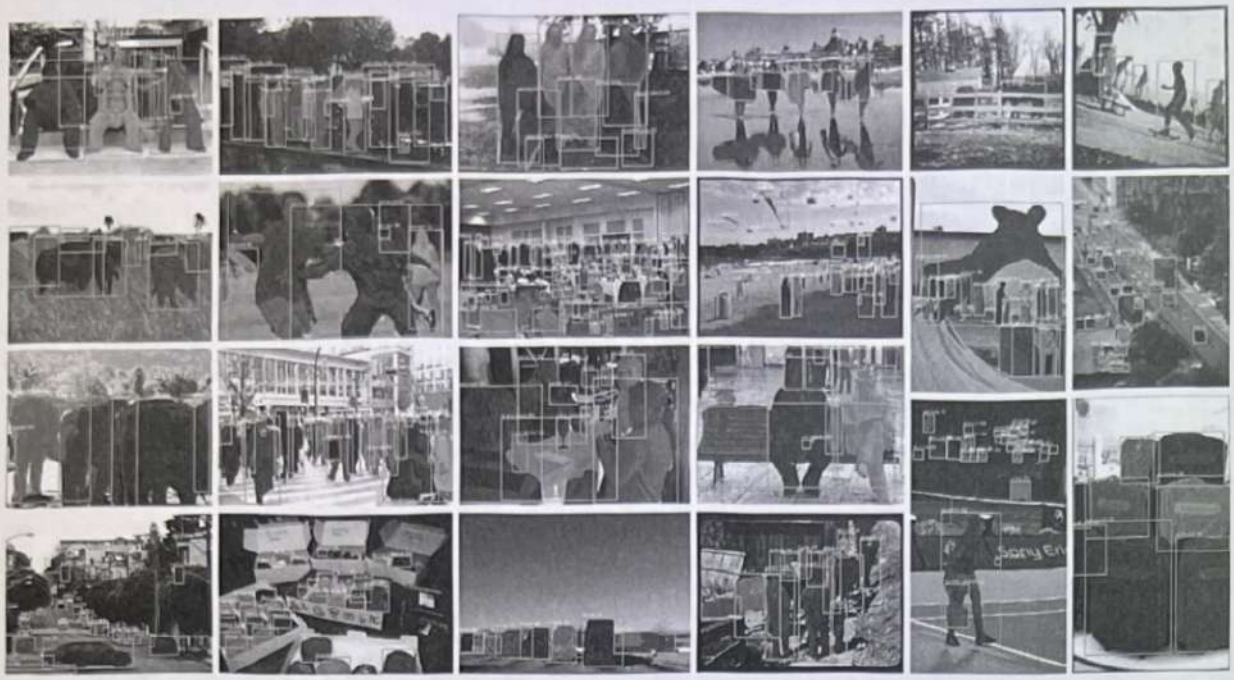

Figure 5. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

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

Table 1. **Instance segmentation** *mask* AP on COCO `test-dev`. MNC [10] and FCIS [26] are the winners of the COCO 2015 and 2016 segmentation challenges, respectively. Without bells and whistles, Mask R-CNN outperforms the more complex FCIS+++, which includes multi-scale train/test, horizontal flip test, and OHEM [38]. All entries are *single-model* results.

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_{50}, AP_{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 with ResNet-101-FPN backbone outperforms FCIS+++ [26], which includes multi-scale train/test, horizontal flip test, and online hard example mining (OHEM) [38]. While outside the scope of this work, we expect many such improvements to be applicable to ours.

Mask R-CNN outputs are visualized in Figures 2 and 5. Mask R-CNN achieves good results even under challenging conditions. In Figure 6 we compare our Mask R-CNN baseline and FCIS+++ [26]. 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.

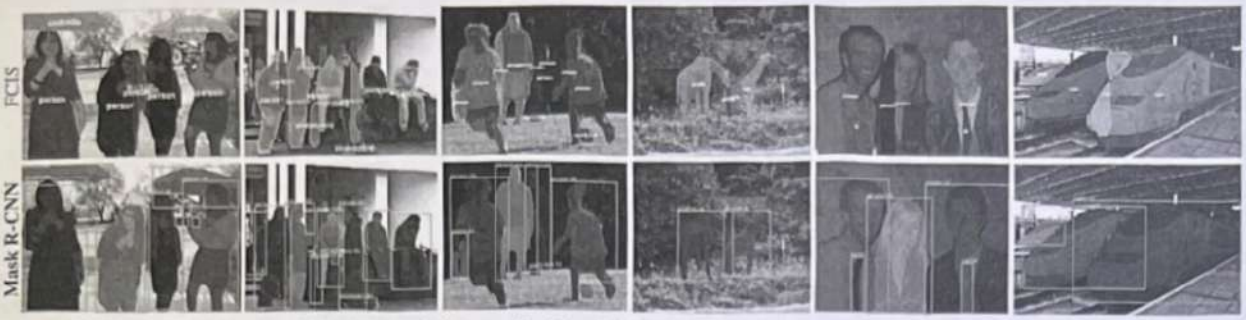

Figure 6. FCIS+++ [26] (top) *vs.* Mask R-CNN (bottom, ResNet-101-FPN). FCIS exhibits systematic artifacts on overlapping objects.

net-depth-features	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}
softmax	24.8	44.1	25.1
sigmoid	**30.3**	**51.2**	**31.5**
	+5.5	*+7.1*	*+6.4*

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

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

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

	AP	AP_{50}	AP_{75}	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
RoIPool	23.6	46.5	21.6	28.2	52.7	26.9
RoIAlign	**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.

	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.

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 (by a per-pixel *sigmoid* and a *binary* loss). In Table 2b, we compare this to using a per-pixel *softmax* and a *multinomial* loss (as commonly used in FCN [30]). 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 *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

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{bb}_{S}	AP^{bb}_{M}	AP^{bb}_{L}
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3	19.8	40.2	48.8
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**	**22.1**	**43.2**	51.2

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

(*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 2e, 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 by "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

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.

	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
CMU-Pose+++ [6]	61.8	84.9	67.5	57.1	68.2
G-RMI [32]†	62.4	84.0	68.5	**59.1**	68.1
Mask R-CNN, keypoint-only	62.7	87.0	68.4	57.4	71.1
Mask R-CNN, keypoint & mask	**63.1**	**87.3**	**68.7**	57.8	**71.4**

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 ~5 points (clarified in personal communication). †: G-RMI was trained on COCO *plus* MPII [1] (25k images), using two models (Inception-ResNet-v2 for bounding box detection and ResNet-101 for keypoints).

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 Figure 4 (right). The keypoint head consists of a stack of eight 3×3 512-d conv layers, followed by a deconv layer and 2× 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.

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 (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 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 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 multi-task 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).

	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**

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).

	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**

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

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 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 be an effective framework for other instance-level tasks.

2015

U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies,
University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

Abstract. There is large consent that successful training of deep networks requires many thousand annotated training samples. In this paper, we present a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. We show that such a network can be trained end-to-end from very few images and outperforms the prior best method (a sliding-window convolutional network) on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks. Using the same network trained on transmitted light microscopy images (phase contrast and DIC) we won the ISBI cell tracking challenge 2015 in these categories by a large margin. Moreover, the network is fast. Segmentation of a 512x512 image takes less than a second on a recent GPU. The full implementation (based on Caffe) and the trained networks are available at http://lmb.informatik.uni-freiburg.de/people/ronneber/u-net.

1 Introduction

In the last two years, deep convolutional networks have outperformed the state of the art in many visual recognition tasks, e.g. [7,3]. While convolutional networks have already existed for a long time [8], their success was limited due to the size of the available training sets and the size of the considered networks. The breakthrough by Krizhevsky et al. [7] was due to supervised training of a large network with 8 layers and millions of parameters on the ImageNet dataset with 1 million training images. Since then, even larger and deeper networks have been trained [12].

The typical use of convolutional networks is on classification tasks, where the output to an image is a single class label. However, in many visual tasks, especially in biomedical image processing, the desired output should include localization, i.e., a class label is supposed to be assigned to each pixel. Moreover, thousands of training images are usually beyond reach in biomedical tasks. Hence, Ciresan et al. [1] trained a network in a sliding-window setup to predict the class label of each pixel by providing a local region (patch) around that pixel

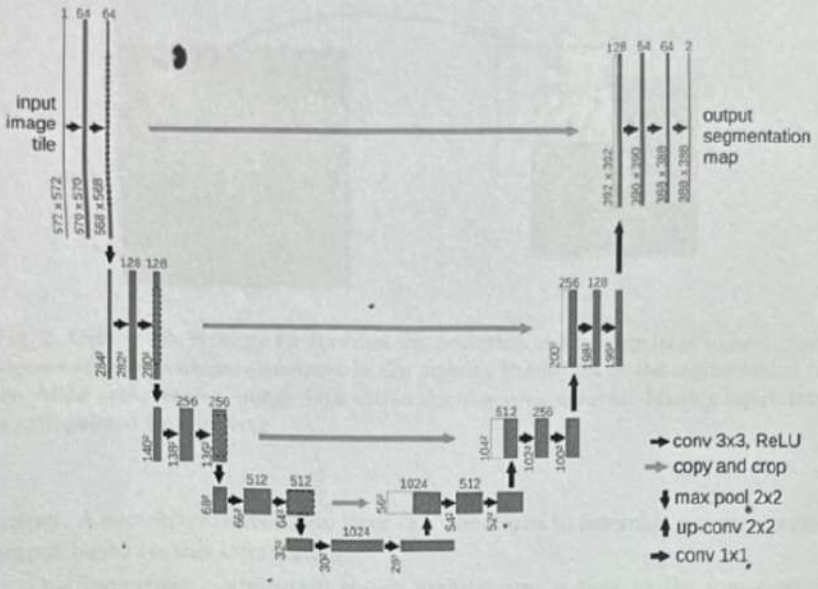

Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

as input. First, this network can localize. Secondly, the training data in terms of patches is much larger than the number of training images. The resulting network won the EM segmentation challenge at ISBI 2012 by a large margin.

Obviously, the strategy in Ciresan et al. [1] has two drawbacks. First, it is quite slow because the network must be run separately for each patch, and there is a lot of redundancy due to overlapping patches. Secondly, there is a trade-off between localization accuracy and the use of context. Larger patches require more max-pooling layers that reduce the localization accuracy, while small patches allow the network to see only little context. More recent approaches [11,4] proposed a classifier output that takes into account the features from multiple layers. Good localization and the use of context are possible at the same time.

In this paper, we build upon a more elegant architecture, the so-called "fully convolutional network" [9]. We modify and extend this architecture such that it works with very few training images and yields more precise segmentations; see Figure 1. The main idea in [9] is to supplement a usual contracting network by successive layers, where pooling operators are replaced by upsampling operators. Hence, these layers increase the resolution of the output. In order to localize, high resolution features from the contracting path are combined with the upsampled

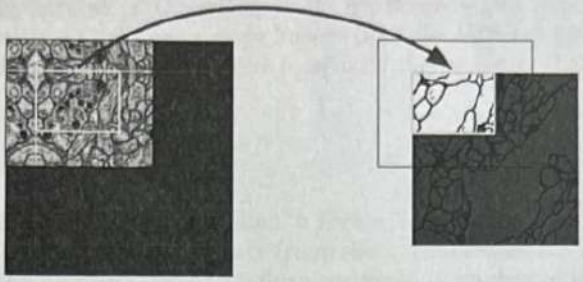

Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

output. A successive convolution layer can then learn to assemble a more precise output based on this information.

One important modification in our architecture is that in the upsampling part we have also a large number of feature channels, which allow the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path, and yields a u-shaped architecture. The network does not have any fully connected layers and only uses the valid part of each convolution, i.e., the segmentation map only contains the pixels, for which the full context is available in the input image. This strategy allows the seamless segmentation of arbitrarily large images by an overlap-tile strategy (see Figure 2). To predict the pixels in the border region of the image, the missing context is extrapolated by mirroring the input image. This tiling strategy is important to apply the network to large images, since otherwise the resolution would be limited by the GPU memory.

As for our tasks there is very little training data available, we use excessive data augmentation by applying elastic deformations to the available training images. This allows the network to learn invariance to such deformations, without the need to see these transformations in the annotated image corpus. This is particularly important in biomedical segmentation, since deformation used to be the most common variation in tissue and realistic deformations can be simulated efficiently. The value of data augmentation for learning invariance has been shown in Dosovitskiy et al. [2] in the scope of unsupervised feature learning.

Another challenge in many cell segmentation tasks is the separation of touching objects of the same class; see Figure 3. To this end, we propose the use of a weighted loss, where the separating background labels between touching cells obtain a large weight in the loss function.

The resulting network is applicable to various biomedical segmentation problems. In this paper, we show results on the segmentation of neuronal structures in EM stacks (an ongoing competition started at ISBI 2012), where we out-

performed the network of Ciresan et al. [1]. Furthermore, we show results for cell segmentation in light microscopy images from the ISBI cell tracking challenge 2015. Here we won with a large margin on the two most challenging 2D transmitted light datasets.

2 Network Architecture

The network architecture is illustrated in Figure 1. It consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

To allow a seamless tiling of the output segmentation map (see Figure 2), it is important to select the input tile size such that all 2x2 max-pooling operations are applied to a layer with an even x- and y-size.

3 Training

The input images and their corresponding segmentation maps are used to train the network with the stochastic gradient descent implementation of Caffe [6]. Due to the unpadded convolutions, the output image is smaller than the input by a constant border width. To minimize the overhead and make maximum use of the GPU memory, we favor large input tiles over a large batch size and hence reduce the batch to a single image. Accordingly we use a high momentum (0.99) such that a large number of the previously seen training samples determine the update in the current optimization step.

The energy function is computed by a pixel-wise soft-max over the final feature map combined with the cross entropy loss function. The soft-max is defined as $p_k(\mathbf{x}) = \exp(a_k(\mathbf{x}))/\left(\sum_{k'=1}^{K} \exp(a_{k'}(\mathbf{x}))\right)$ where $a_k(\mathbf{x})$ denotes the activation in feature channel k at the pixel position $\mathbf{x} \in \Omega$ with $\Omega \subset \mathbb{Z}^2$. K is the number of classes and $p_k(\mathbf{x})$ is the approximated maximum-function. I.e. $p_k(\mathbf{x}) \approx 1$ for the k that has the maximum activation $a_k(\mathbf{x})$ and $p_k(\mathbf{x}) \approx 0$ for all other k. The cross entropy then penalizes at each position the deviation of $p_{\ell(\mathbf{x})}(\mathbf{x})$ from 1 using

$$E = \sum_{\mathbf{x} \in \Omega} w(\mathbf{x}) \log(p_{\ell(\mathbf{x})}(\mathbf{x})) \tag{1}$$

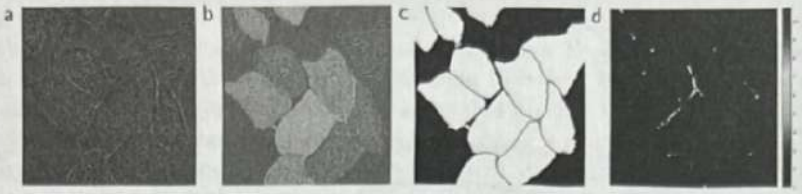

Fig. 3. HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (**a**) raw image. (**b**) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (**c**) generated segmentation mask (white: foreground, black: background). (**d**) map with a pixel-wise loss weight to force the network to learn the border pixels.

where $\ell : \Omega \to \{1, \dots, K\}$ is the true label of each pixel and $w : \Omega \to \mathbb{R}$ is a weight map that we introduced to give some pixels more importance in the training.

We pre-compute the weight map for each ground truth segmentation to compensate the different frequency of pixels from a certain class in the training data set, and to force the network to learn the small separation borders that we introduce between touching cells (See Figure 3c and d).

The separation border is computed using morphological operations. The weight map is then computed as

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right) \tag{2}$$

where $w_c : \Omega \to \mathbb{R}$ is the weight map to balance the class frequencies, $d_1 : \Omega \to \mathbb{R}$ denotes the distance to the border of the nearest cell and $d_2 : \Omega \to \mathbb{R}$ the distance to the border of the second nearest cell. In our experiments we set $w_0 = 10$ and $\sigma \approx 5$ pixels.

In deep networks with many convolutional layers and different paths through the network, a good initialization of the weights is extremely important. Otherwise, parts of the network might give excessive activations, while other parts never contribute. Ideally the initial weights should be adapted such that each feature map in the network has approximately unit variance. For a network with our architecture (alternating convolution and ReLU layers) this can be achieved by drawing the initial weights from a Gaussian distribution with a standard deviation of $\sqrt{2/N}$, where N denotes the number of incoming nodes of one neuron [5]. E.g. for a 3x3 convolution and 64 feature channels in the previous layer $N = 9 \cdot 64 = 576$.

3.1 Data Augmentation

Data augmentation is essential to teach the network the desired invariance and robustness properties, when only few training samples are available. In case of

microscopical images we primarily need shift and rotation invariance as well as robustness to deformations and gray value variations. Especially random elastic deformations of the training samples seem to be the key concept to train a segmentation network with very few annotated images. We generate smooth deformations using random displacement vectors on a coarse 3 by 3 grid. The displacements are sampled from a Gaussian distribution with 10 pixels standard deviation. Per-pixel displacements are then computed using bicubic interpolation. Drop-out layers at the end of the contracting path perform further implicit data augmentation.

4 Experiments

We demonstrate the application of the u-net to three different segmentation tasks. The first task is the segmentation of neuronal structures in electron microscopic recordings. An example of the data set and our obtained segmentation is displayed in Figure 2. We provide the full result as Supplementary Material. The data set is provided by the EM segmentation challenge [14] that was started at ISBI 2012 and is still open for new contributions. The training data is a set of 30 images (512x512 pixels) from serial section transmission electron microscopy of the Drosophila first instar larva ventral nerve cord (VNC). Each image comes with a corresponding fully annotated ground truth segmentation map for cells (white) and membranes (black). The test set is publicly available, but its segmentation maps are kept secret. An evaluation can be obtained by sending the predicted membrane probability map to the organizers. The evaluation is done by thresholding the map at 10 different levels and computation of the "warping error", the "Rand error" and the "pixel error" [14].

The u-net (averaged over 7 rotated versions of the input data) achieves without any further pre- or postprocessing a warping error of 0.0003529 (the new best score, see Table 1) and a rand-error of 0.0382.

This is significantly better than the sliding-window convolutional network result by Ciresan et al. [1], whose best submission had a warping error of 0.000420 and a rand error of 0.0504. In terms of rand error the only better performing

Table 1. Ranking on the EM segmentation challenge [14] (march 6th, 2015), sorted by warping error.

Rank	Group name	Warping Error	Rand Error	Pixel Error
	** human values **	0.000005	0.0021	0.0010
1.	u-net	**0.000353**	0.0382	0.0611
2.	DIVE-SCI	0.000355	0.0305	0.0584
3.	IDSIA [1]	0.000420	0.0504	0.0613
4.	DIVE	0.000430	0.0545	**0.0582**
⋮				
10.	IDSIA-SCI	0.000653	**0.0189**	0.1027

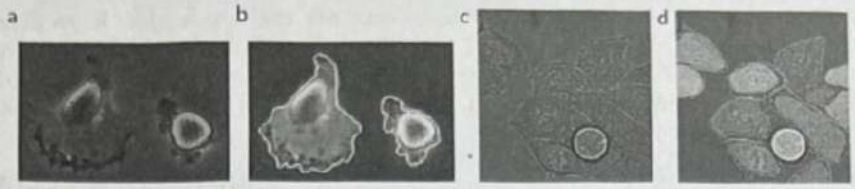

Fig. 4. Result on the ISBI cell tracking challenge. (**a**) part of an input image of the "PhC-U373" data set. (**b**) Segmentation result (cyan mask) with manual ground truth (yellow border) (**c**) input image of the "DIC-HeLa" data set. (**d**) Segmentation result (random colored masks) with manual ground truth (yellow border).

Table 2. Segmentation results (IOU) on the ISBI cell tracking challenge 2015.

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

algorithms on this data set use highly data set specific post-processing methods[1] applied to the probability map of Ciresan et al. [1].

We also applied the u-net to a cell segmentation task in light microscopic images. This segmenation task is part of the ISBI cell tracking challenge 2014 and 2015 [10,13]. The first data set "PhC-U373"[2] contains Glioblastoma-astrocytoma U373 cells on a polyacrylimide substrate recorded by phase contrast microscopy (see Figure 4a,b and Supp. Material). It contains 35 partially annotated training images. Here we achieve an average IOU ("intersection over union") of 92%, which is significantly better than the second best algorithm with 83% (see Table 2). The second data set "DIC-HeLa"[3] are HeLa cells on a flat glass recorded by differential interference contrast (DIC) microscopy (see Figure 3, Figure 4c,d and Supp. Material). It contains 20 partially annotated training images. Here we achieve an average IOU of 77.5% which is significantly better than the second best algorithm with 46%.

5 Conclusion

The u-net architecture achieves very good performance on very different biomedical segmentation applications. Thanks to data augmentation with elastic defor-

[1] The authors of this algorithm have submitted 78 different solutions to achieve this result.

[2] Data set provided by Dr. Sanjay Kumar. Department of Bioengineering University of California at Berkeley. Berkeley CA (USA)

[3] Data set provided by Dr. Gert van Cappellen Erasmus Medical Center. Rotterdam. The Netherlands