Conditional Convolutions for Instance Segmentation

Zhi Tian Chunhua Shen* Hao Chen

The University of Adelaide, Australia

Abstract

We propose a simple yet effective instance segmentation framework, termed CondInst (conditional convolutions for instance segmentation). Top-performing instance segmentation methods such as Mask R-CNN rely on ROI operations (typically ROIPool or ROIAlign) to obtain the final instance masks. In contrast, we propose to solve instance segmentation from a new perspective. Instead of using instance-wise ROIs as inputs to a network of fixed weights, we employ dynamic instance-aware networks, conditioned on instances. CondInst enjoys two advantages: 1) Instance segmentation is solved by a fully convolutional network, eliminating the need for ROI cropping and feature alignment. 2) Due to the much improved capacity of dynamically-generated conditional convolutions, the mask head can be very compact (e.g., 3 conv. layers, each having only 8 channels), leading to significantly faster inference. We demonstrate a simpler instance segmentation method that can achieve improved performance in both accuracy and inference speed. On the COCO dataset, we outperform a few recent methods including well-tuned Mask R-CNN baselines, without longer training schedules needed. Code is available: `https://git.io/AdelaiDet`

Keywords: Conditional convolutions, instance segmentation

1. Introduction

Instance segmentation is a fundamental yet challenging task in computer vision, which requires an algorithm to predict a per-pixel mask with a category label for each instance of interest in an image. Despite a few works being proposed recently, the dominant framework for instance segmentation is still the two-stage method Mask R-CNN [3], which casts instance segmentation into a two-stage detection-and-segmentation task. Mask R-CNN first employs an object detector Faster R-CNN to predict a bounding-box for each instance. Then for each instance, regions-of-interest (ROIs) are cropped from the networks' feature maps using the ROIAlign operation. To predict the final masks for each instance, a compact fully convolutional network (FCN) (*i.e.*, mask head) is applied to these ROIs to perform foreground/background segmentation. However, this ROI-based method may have the following drawbacks. 1) Since ROIs are often axis-aligned bounding-boxes, for objects with irregular shapes, they may contain an excessive amount of irrelevant image content including background and other instances. This issue may be mitigated by using rotated ROIs, but with the price of a more complex pipeline. 2) In order to distinguish between the foreground instance and the background stuff or instance(s), the mask head requires a relatively larger receptive field to encode sufficiently large context information. As a result, a stack of 3×3 convolutions is needed in the mask head (*e.g.*, four 3×3 convolutions with 256 channels in Mask R-CNN). It considerably increases computational complexity of the mask head, resulting that the inference time significantly varies in the number of instances. 3) ROIs are typically of

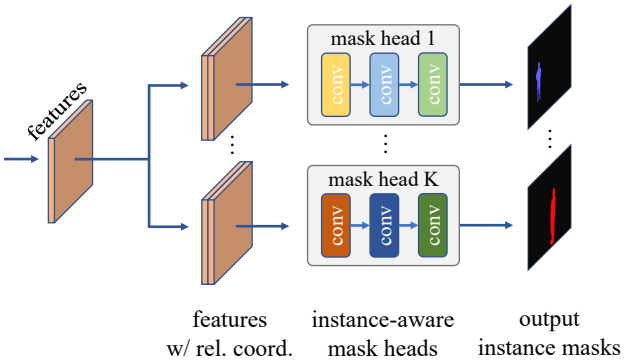

Figure 1. CondInst makes use of instance-aware mask heads to predict the masks for each instance. K is the number of instances to be predicted. The filters in the mask head vary with different instances, which are dynamically-generated and conditioned on the target instance. For the non-last conv. layers in the mask head, ReLU is used as the activation function and no normalization layer such as batch normalization [1] is used here. The last conv. layer uses sigmoid to predict the probability of being mask foreground.

*Corresponding author, e-mail: `chunhua.shen@adelaide.edu.au`

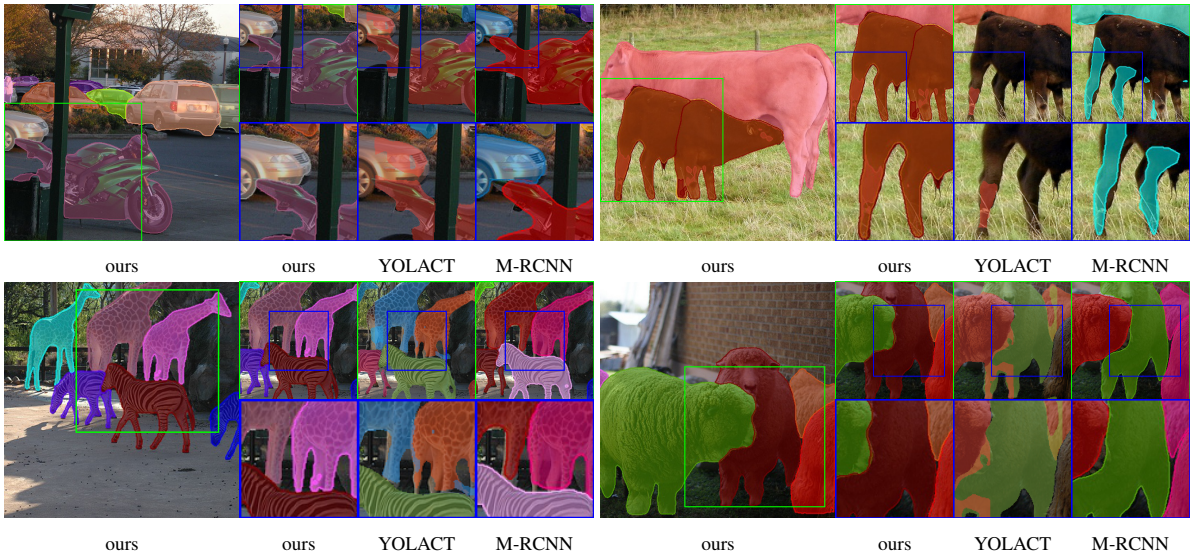

Figure 2. Qualitative comparisons with other methods. We compare the proposed CondInst against YOLACT [2] and Mask R-CNN [3]. Our masks are generally of higher quality (*e.g.*, preserving more details). Best viewed on screen.

different sizes. In order to use effective batched computation in modern deep learning frameworks [4, 5], a resizing operation is often required to resize the cropped regions into patches of the same size. For instance, Mask R-CNN resizes all the cropped regions to 14×14 (upsampled to 28×28 using a deconvolution), which restricts the output resolution of instance segmentation, as large instances would require higher resolutions to retain details at the boundary.

In computer vision, the closest task to instance segmentation is semantic segmentation, for which fully convolutional networks (FCNs) have shown dramatic success [6, 7]. FCNs also have shown excellent performance on many other per-pixel prediction tasks ranging from low-level image processing such as denoising, super-resolution; to mid-level tasks such as optical flow estimation and contour detection; and high-level tasks including recent single-shot object detection [8], monocular depth estimation [9] and counting [10]. However, almost all the instance segmentation methods based on FCNs[1] lag behind state-of-the-art ROI-based methods. Why do the versatile FCNs perform unsatisfactorily on instance segmentation? We observe that the major difficulty of applying FCNs to instance segmentation is that the similar image appearance may require different predictions but FCNs struggle at achieving this. For example, if two persons A and B with the similar appearance are in an input image, when predicting the instance mask of A, the FCN needs to predict B as background w.r.t. A, which can be difficult as they look similar in appearance. Therefore, the ROI operation is used to crop the person of interest, *e.g.*, A; and filter out B. Essentially, instance segmentation needs two types of information: 1) *appearance* information to categorize objects; and 2) *location* information to distinguish multiple objects belonging to the same category. Almost all methods rely on ROI cropping, which explicitly encodes the location information of instances. In contrast, CondInst exploits the location information by using location/instance-sensitive convolution filters as well as relative coordinates that are appended to the feature map.

Thus, we advocate a new solution that uses instance-aware FCNs for instance mask prediction. In other words, instead of using a standard ConvNet with a fixed set of convolutional filters as the mask head for predicting all instances, the network parameters are adapted according to the instance to be predicted. Inspired by dynamic filtering networks [11] and CondConv [12], for each instance, a controller sub-network (see Fig. 3) dynamically generates the mask FCN network parameters (conditioned on the center area of the instance), which is then used to predict the mask of this instance. It is expected that the network parameters can encode the characteristics of this instance, and only fires on the pixels of this instance, which thus bypasses the difficulty mentioned above. These conditional mask heads are applied to the whole feature maps, *eliminating the need for ROI operations*. At the first glance, the idea may not work well as instance-wise mask heads may incur a large number of network parameters provided that some images contain as many as dozens of instances. However, we show that a very compact FCN mask head with dynamically-generated filters can already outperform previous ROI-based Mask R-CNN, resulting in much reduced computational complexity

[1] By FCNs, we mean the vanilla FCNs in [6] that only involve convolutions and pooling.

per instance than that of the mask head in Mask R-CNN.

We summarize our main contributions as follow.

- We attempt to solve instance segmentation from a new perspective. To this end, we propose the CondInst instance segmentation framework, which achieves improved instance segmentation performance than existing methods such as Mask R-CNN while being faster. To our knowledge, this is the first time that a new instance segmentation framework outperforms recent state-of-the-art both in accuracy and speed.
- CondInst is fully convolutional and avoids the aforementioned resizing operation in many existing methods, as CondInst does not rely on ROI operations. Without having to resize feature maps leads to high-resolution instance masks with more accurate edges.
- Unlike previous methods, in which the filters in its mask head are fixed for all the instances once trained, the filters in our mask head are dynamically generated and conditioned on instances. As the filters are only asked to predict the mask of only one instance, it largely eases the learning requirement and thus reduces the load of the filters. As a result, the mask head can be extremely light-weight, significantly reducing the inference time per instance. Compared with the bounding box detector FCOS, CondInst needs only $\sim$10% more computational time, even processing the maximum number of instances per image (*i.e.*, 100 instances).
- Even without resorting to longer training schedules as needed in recent works [2, 13], CondInst achieves state-of-the-art performance while being faster in inference. We hope that CondInst can be a new strong alternative to popular methods such as Mask R-CNN for the instance segmentation task.

Moreover, CondInst can be immediately applied to panoptic segmentation due to its flexible design. We believe that with minimal re-design effort, the proposed CondInst can be used to solve all instance-level recognition tasks that were previously solved with an ROI-based pipeline.

1.1. Related Work

Here we review some work that is most relevant to ours.

Conditional Convolutions. Unlike traditional convolutional layers, which have fixed filters once trained, the filters of conditional convolutions are conditioned on the input and are dynamically generated by another network (*i.e.*, a controller). This idea has been explored previously in dynamic filter networks [11] and CondConv [12] mainly for the purpose of increasing the capacity of a classification network. In this work, we extend this idea to solve the significantly more challenging task of instance segmentation.

Instance Segmentation. To date, the dominant framework for instance segmentation is still Mask R-CNN. Mask R-CNN first employs an object detector to detect the bounding-boxes of instances (*e.g.*, ROIs). With these bounding-boxes, an ROI operation is used to crop the features of the instance from the feature maps. Finally, a compact FCN head is used to obtain the desired instance masks. Many works [14–16] with top performance are built on Mask R-CNN. Moreover, some works have explored to apply FCNs to instance segmentation. InstanceFCN [17] may be the first instance segmentation method that is fully convolutional. InstanceFCN proposes to predict position-sensitive score maps with vanilla FCNs. Afterwards, these score maps are assembled to obtain the desired instance masks. Note that InstanceFCN does not work well with overlapping instances. Others [18–20] attempt to first perform segmentation and the desired instance masks are formed by assembling the pixels of the same instance. To our knowledge, thus far none of these methods can outperform Mask R-CNN both in accuracy and speed on the public COCO benchmark dataset.

The recent YOLACT [2] and BlendMask [21] may be viewed as a reformulation of Mask RCNN, which decouple ROI detection and feature maps used for mask prediction. Wang et al. developed a simple FCN based instance segmentation method, showing competitive performance [22]. PolarMask developed a new simple mask representation for instance segmentation [23], which extends the bounding box detector FCOS [8].

Recently AdaptIS [24] proposes to solve panoptic segmentation with FiLM [25]. The idea shares some similarity with CondInst in that information about an instance is encoded in the coefficients generated by FiLM. Since only the batch normalization coefficients are dynamically generated, AdaptIS needs a large mask head to achieve good performance. In contrast, CondInst directly encodes them into conv. filters of the mask head, thus having much stronger capacity. As a result, even with a very compact mask head, we believe that CondInst can achieve instance segmentation accuracy that would not be possible for AdaptIS to attain.

2. Instance Segmentation with CondInst

2.1. Overall Architecture

Given an input image $I \in \mathbb{R}^{H \times W \times 3}$, the goal of instance segmentation is to predict the pixel-level mask and the category of each instance of interest in the image. The ground-truth of instance segmentation are defined as $\{(M_i, c_i)\}$, where $M_i \in \{0,1\}^{H \times W}$ is the mask for the i-th instance and $c_i \in \{1, 2, ..., C\}$ is the category. C is 80 on MS-COCO [27]. Unlike semantic segmentation, which only requires to predict one mask for an input image, instance segmentation needs to predict a variable number of masks,

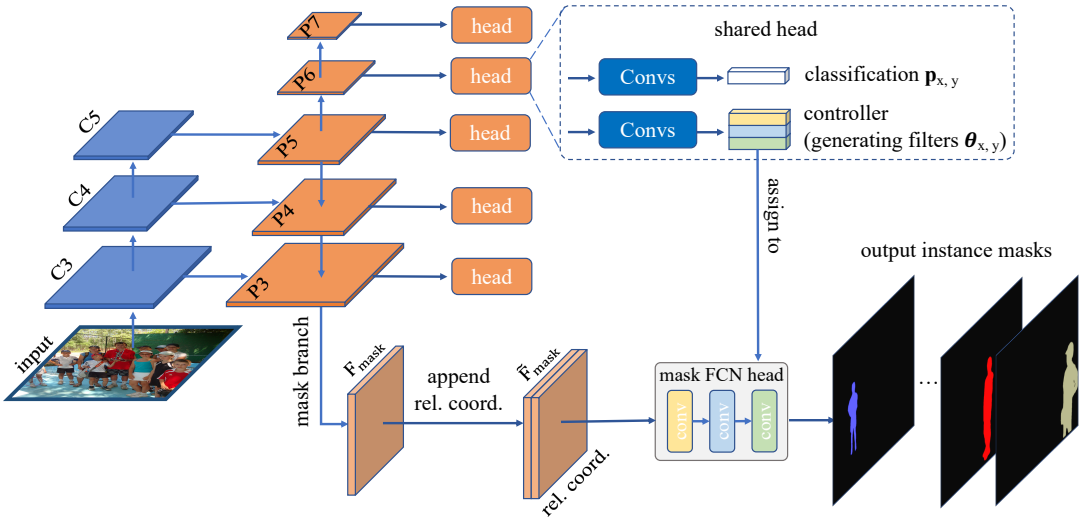

Figure 3. The overall architecture of CondInst. C_3, C_4 and C_5 are the feature maps of the backbone network (*e.g.*, ResNet-50). P_3 to P_7 are the FPN feature maps as in [8, 26]. $\mathbf{F}_{mask}$ is the mask branch's output and $\tilde{\mathbf{F}}_{mask}$ is obtained by concatenating the relative coordinates to $\mathbf{F}_{mask}$. The classification head predicts the class probability $\boldsymbol{p}_{x,y}$ of the target instance at location (x, y), same as in FCOS. Note that the classification and conv. parameter generating heads (in the dashed box) are applied to $P_3 \cdots P_7$. The mask head is instance-aware, whose conv. filters $\boldsymbol{\theta}_{x,y}$ are dynamically generated for each instance, and is applied to $\tilde{\mathbf{F}}_{mask}$ as many times as the number of instances in the image (refer to Fig. 1).

depending on the number of instances in the image. This poses a challenge when applying traditional FCNs [6] to instance segmentation. In this work, our core idea is that for an image with K instances, K different mask heads will be dynamically generated, and each mask head will contain the characteristics of its target instance in their filters. As a result, when the mask is applied to an input, it will only fire on the pixels of the instance, thus producing the mask prediction of the instance. We illustrate the process in Fig. 1.

Recall that Mask R-CNN employs an object detector to predict the bounding-boxes of the instances in the input image. The bounding-boxes are actually the way that Mask R-CNN represents instances. Similarly, CondInst employs the instance-aware filters to represent the instances. In other words, instead of encoding the instance concept into the bounding-boxes, CondInst implicitly encodes it into the parameters of the mask heads, which is a much more flexible way. For example, it can easily represent the irregular shapes that are hard to be tightly enclosed by a bounding-box. This is one of CondInst's advantages over the previous ROI-based methods.

Similar to the way that ROI-based methods obtain bounding-boxes, the instance-aware filters can also be obtained with an object detector. In this work, we build CondInst on the popular object detector FCOS [8] due to its simplicity and flexibility. Also, the elimination of anchor-boxes in FCOS can also save the number of parameters and the amount of computation of CondInst. As shown in Fig. 3, following FCOS [8], we make use of the feature maps $\{P_3, P_4, P_5, P_6, P_7\}$ of feature pyramid networks (FPNs) [26], whose down-sampling ratios are 8, 16, 32, 64 and 128, respectively. As shown in Fig. 3, on each feature level of the FPN, some functional layers (in the dash box) are applied to make instance-related predictions. For example, the class of the target instance and the dynamically-generated filters for the instance. In this sense, CondInst can be viewed as the same as Mask R-CNN, both of which first attend to instances in an image and then predict the pixel-level masks of the instances (*i.e.*, instance-first).

Besides the detector, as shown in Fig. 3, there is also a mask branch, which provides the feature maps that our generated mask heads take as inputs to predict the desired instance mask. The feature maps are denoted by $\mathbf{F}_{mask} \in \mathbb{R}^{H_{mask} \times W_{mask} \times C_{mask}}$. The mask branch is connected to FPN level P_3 and thus its output resolution is $\frac{1}{8}$ of the input image resolution. The mask branch has four 3×3 convolutions with 128 channels before the last layer. Afterwards, in order to reduce the number of the generated parameters, the last layer of the mask branch reduces the number of channels from 128 to 8 (*i.e.*, $C_{mask} = 8$). Surprisingly, using $C_{mask} = 8$ can already achieve superior performance and using a larger C_{mask} here (*e.g.*, 16) cannot improve the performance, as shown in our experiments. Even more aggressively, using $C_{mask} = 2$ only degrades the performance by

$\sim 0.3\%$ in mask AP. Moreover, as shown in Fig. 3, $\mathbf{F}_{mask}$ is combined with a map of the coordinates, which are relative coordinates from all the locations on $\mathbf{F}_{mask}$ to the location (x, y) (*i.e.*, where the filters of the mask head are generated). Then, the combination is sent to the mask head to predict the instance mask. The relative coordinates provide a strong cue for predicting the instance mask, as shown in our experiments. Moreover, a single sigmoid is used as the final output of the mask head, and thus the mask prediction is class-agnostic. The class of the instance is predicted by the classification head in parallel with the controller, as shown in Fig. 3.

The resolution of the original mask prediction is same as the resolution of F_{mask}, which is $\frac{1}{8}$ of the input image resolution. In order to produce high-resolution instance masks, a bilinear upsampling is used to upsample the mask prediction by 4, resulting in 400×512 mask prediction (if the input image size is 800×1024). We will show that the upsampling is crucial to the final instance segmentation performance of CondInst in experiments. Note that the mask's resolution is much higher than that of Mask R-CNN (only 28×28 as mentioned before).

2.2. Network Outputs and Training Targets

Similar to FCOS, each location on the FPN's feature maps P_i either is associated with an instance, thus being a positive sample, or is considered a negative sample. The associated instance and label for each location are determined as follows. Let us consider the feature maps $P_i \in \mathbb{R}^{H \times W \times C}$ and let s be its down-sampling ratio. As shown in previous works [8, 28, 29], a location (x, y) on the feature maps can be mapped back onto the input image as $(\lfloor \frac{s}{2} \rfloor + xs, \lfloor \frac{s}{2} \rfloor + ys)$. If the mapped location falls in the center region of an instance, the location is considered to be responsible for the instance. Any locations outside the center regions are labeled as negative samples. The center region is defined as the box $(c_x - rs, c_y - rs, c_x + rs, c_y + rs)$, where (c_x, c_y) denotes the mass center of the instance, s is the down-sampling ratio of P_i and r is a constant scalar being 1.5 as in FCOS [8]. As shown in Fig. 3, at a location (x, y) on P_i, CondInst has the following output heads.

Classification Head. The classification head predicts the class of the instance associated with the location. The ground-truth target is the instance's class c_i or 0 (*i.e.*, background). As in FCOS, the network predicts a C-D vector $\boldsymbol{p}_{x,y}$ for the classification and each element in $\boldsymbol{p}_{x,y}$ corresponds to a binary classifier, where C is the number of categories.

Controller Head. The controller head, which has the same architecture as the above classification head, is used to generate the parameters of the conv. filters for the instance at the location. As mentioned before, these generated filters are used in the mask head to predict the mask of this particular instance. This is the core contribution of our work.

To predict the filters, we concatenate all the parameters of the filters (*i.e.*, weights and biases) together as an N-D vector $\boldsymbol{\theta}_{x,y}$, where N is the total number of the parameters. Thus, the controller head has N output channels. As mentioned before, using a very few parameters (*e.g.*, 169 parameters), CondInst can already achieve excellent instance segmentation performance, which not only makes the parameters can be easily generated but also results in a mask head with low computational complexity. Thus, we use a very compact FCN as the mask head, which has three 1×1 convolutions, each having 8 channels and using ReLU as the activation function except for the last one. No normalization layer such as batch normalization [1] is used here. The last layer has 1 output channel and uses sigmoid to predict the probability of being foreground. The generated N-D vector will be reinterpreted into the weights and biases of these filters. As mentioned before, the generated filters contain information about the instance at the location, and thus the mask head with the filters will ideally only fire on the pixels of the instance, even taking as the input the whole feature maps.

Center-ness Head. The center-ness head predicts a scalar depicting the deviation from the location to the center of the target instance. The center-ness score is multiplied with the classification scores and used in NMS remove duplicated detections. We refer readers to FCOS [8] for the details.

Conceptually, CondInst with the above heads can already solve the instance segmentation task since CondInst needs no ROIs. However, we find that if we make use of box-based NMS, the inference time will be much reduced. Thus, we still predict bounding-boxes in CondInst. Following FCOS, CondInst also predicts a 4-D vector $\boldsymbol{t} = (l, t, r, b)$ depicting the distances from the location to four sides (*i.e.*, left, top, right and bottom) of the instance's bounding-box. The ground-truth bounding-boxes can be easily computed from the instance's mask annotation M_i, and thus predicting bounding-boxes introduces no any extra annotations. We would like to highlight that the predicted bounding-boxes are *only* used in NMS and do not involve any ROI operations. Moreover, as shown in Table 5, the bounding-boxes prediction can be removed if the NMS using no bounding-box (*e.g.*, mask NMS or peak NMS [30]) used. This is fundamentally different from previous ROI-based methods, in which the bounding-box prediction is mandatory.

2.3. Loss Function

Formally, the overall loss function of CondInst can be formulated as,

$$L_{overall} = L_{fcos} + \lambda L_{mask}, \tag{1}$$

where L_{fcos} and L_{mask} denote the original loss of FCOS and the loss for instance masks, respectively. λ being 1 in

this work is used to balance the two losses. We refer readers to FCOS for the details of L_{fcos}. L_{mask} is defined as,

$$L_{mask}(\{\boldsymbol{\theta}_{x,y}\}) = \frac{1}{N_{\text{pos}}} \sum_{x,y} \mathbb{1}_{\{c^*_{x,y}>0\}} L_{dice}(MaskHead(\tilde{\mathbf{F}}_{x,y}; \boldsymbol{\theta}_{x,y}), \mathbf{M}^*_{x,y}), \tag{2}$$

where $c^*_{x,y}$ is the classification label of location (x, y), which is the class of the instance associated with the location or 0 (*i.e.*, background) if the location is not associated with any instance. N_{pos} is the number of locations where $c^*_{x,y} > 0$. $\mathbb{1}_{\{c^*_{x,y}>0\}}$ is the indicator function, being 1 if $c^*_{x,y} > 0$ and 0 otherwise. $\boldsymbol{\theta}_{x,y}$ is the generated filters' parameters at location (x, y). $\tilde{\mathbf{F}}_{x,y} \in \mathbb{R}^{H_{mask} \times W_{mask} \times (C_{mask}+2)}$ is the combination of $\mathbf{F}_{mask}$ and a map of coordinates $\mathbf{O}_{x,y} \in \mathbb{R}^{H_{mask} \times W_{mask} \times 2}$. As described before, $\mathbf{O}_{x,y}$ is the relative coordinates from all the locations on $\mathbf{F}_{mask}$ to (x, y) (*i.e.*, the location where the filters are generated). $MaskHead$ denotes the mask head, which consists of a stack of convolutions with dynamic parameters $\boldsymbol{\theta}_{x,y}$. $\mathbf{M}^*_{x,y} \in \{0, 1\}^{H \times W \times C}$ is the mask of the instance associated with location (x, y). L_{dice} is the dice loss as in [31], which is used to overcome the foreground-background sample imbalance. We do not employ focal loss here as it requires special initialization, which cannot be realized if the parameters are dynamically generated. Note that, in order to compute the loss between the predicted mask and the ground-truth mask $\mathbf{M}^*_{x,y}$, they are required to have the same size. As mentioned before, the prediction is upsampled by 4 and thus the final prediction has half the ground-truth mask's resolution. Thus, we downsample $\mathbf{M}^*_{x,y}$ by 2 to make their sizes equal. These operations are omitted in Eq. (2) for clarification.

Moreover, as shown in YOLACT [2], the instance segmentation task can benefit from a joint semantic segmentation task. Thus, we also conduct experiments with the joint semantic segmentation task. However, unless explicitly specified, all the experiments in the paper are *without* the semantic segmentation task. If used, the semantic segmentation loss is added to $L_{overall}$.

2.4. Inference

Given an input image, we forward it through the network to obtain the outputs including classification confidence $\boldsymbol{p}_{x,y}$, center-ness scores, box prediction $\boldsymbol{t}_{x,y}$ and the generated parameters $\boldsymbol{\theta}_{x,y}$. We first follow the steps in FCOS to obtain the bounding-box detections. Afterwards, box-based NMS with the threshold being 0.6 is used to remove duplicated detections and then the top 100 bounding-boxes (*i.e.*, instances) are used to compute masks. Let us assume that K bounding-boxes remain after the process and thus we have K groups of the generated filters. The K groups of filters in turn are used in the mask head. These instance-specific mask heads are applied, in the fashion of FCNs, to the $\tilde{\mathbf{F}}_{x,y}$ (*i.e.*, the combination of $\mathbf{F}_{mask}$ and $\mathbf{O}_{x,y}$) to predict the masks of the instances. Since the mask head is a very compact network (three 1×1 convolutions with 8 channels and 169 parameters in total), the overhead of computing masks is extremely small. For example, even with 100 detections (*i.e.*, the maximum number of detections per image on MS-COCO), only less 5 milliseconds in total are spent on the mask heads, which only adds $\sim 10\%$ computational time to the base detector FCOS. In contrast, the mask head of Mask R-CNN has four 3×3 convolutions with 256 channels, thus having more than 2.3M parameters and taking longer computational time.

3. Experiments

We evaluate CondInst on the large-scale benchmark MS-COCO [27]. Following the common practice [3, 8, 32], our models are trained with split `train2017` (115K images) and all the ablation experiments are evaluated on split `val2017` (5K images). Our main results are reported on the `test-dev` split (20K images).

3.1. Implementation Details

Unless specified, we make use of the following implementation details. Following FCOS [8], ResNet-50 [33] is used as our backbone network and the weights pre-trained on ImageNet [34] are used to initialize it. For the newly added layers, we initialize them as in [8]. Our models are trained with stochastic gradient descent (SGD) over 8 V100 GPUs for 90K iterations with the initial learning rate being 0.01 and a mini-batch of 16 images. The learning rate is reduced by a factor of 10 at iteration $60K$ and $80K$, respectively. Weight decay and momentum are set as 0.0001 and 0.9, respectively. Following Detectron2 [35], the input images are resized to have their shorter sides in $[640, 800]$ and their longer sides less or equal to 1333 during training. Left-right flipping data augmentation is also used during training. When testing, we do not use any data augmentation and only the scale of the shorter side being 800 is used. The inference time in this work is measured on a single V100 GPU with 1 image per batch.

3.2. Architectures of the Mask Head

In this section, we discuss the design choices of the mask head in CondInst. To our surprise, the performance is insensitive to the architectures of the mask head. Our baseline is the mask head of three 1×1 convolutions with 8 channels (*i.e.*, width $= 8$). As shown in Table 1 (3rd row), it achieves 35.7% in mask AP. Next, we first conduct experiments by varying the depth of the mask head. As shown in Table 1a, apart from the mask head with depth being 1, all other mask heads (*i.e.*, depth $= 2, 3$ and 4) attain similar performance.

depth	time	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	**2.2**	30.9	52.9	31.4	14.0	33.3	45.1
2	3.3	35.5	56.1	37.8	17.0	38.9	50.8
3	4.5	**35.7**	**56.3**	37.8	17.1	**39.1**	50.2
4	5.6	**35.7**	56.2	**37.9**	**17.2**	38.7	**51.5**

(a) Varying the depth (width $= 8$).

width	time	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
2	**2.5**	34.1	55.4	35.8	15.9	37.2	49.1
4	2.6	35.6	**56.5**	**38.1**	17.0	**39.2**	**51.4**
8	4.5	**35.7**	**56.3**	37.8	17.1	39.1	50.2
16	4.7	35.6	56.2	37.9	**17.2**	38.8	50.8

(b) Varying the width (depth $= 3$).

Table 1: Instance segmentation results with different architectures of the mask head on MS-COCO `val2017` split. "depth": the number of layers in the mask head. "width": the number of channels of these layers. "time": the milliseconds that the mask head takes for processing 100 instances.

C_{mask}	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	34.8	55.9	36.9	16.7	38.0	50.1
2	35.4	56.2	37.6	16.9	38.9	50.4
4	35.5	56.2	**37.9**	17.0	39.0	50.8
8	**35.7**	**56.3**	37.8	**17.1**	**39.1**	50.2
16	35.5	56.1	37.7	16.4	**39.1**	**51.2**

Table 2: The instance segmentation results by varying the number of channels of the mask branch output (*i.e.*, C_{mask}) on MS-COCO `val2017` split. As shown in the table, the performance keeps almost the same if C_{mask} is in a reasonable range, which suggests that CondInst is robust to the design choice.

The mask head with depth being 1 achieves inferior performance as in this case the mask head is actually a linear mapping, which has overly weak capacity. Moreover, as shown in Table 1b, varying the width (*i.e.*, the number of the channels) does not result in a remarkable performance change either as long as the width is in a reasonable range. We also note that our mask head is extremely light-weight as the filters in our mask head are dynamically generated. As shown in Table 1, our baseline mask head only takes 4.5 ms per 100 instances (the maximum number of instances on MS-COCO), which suggests that our mask head only adds small computational overhead to the base detector. Moreover, our baseline mask head only has 169 parameters in total. In sharp contrast, the mask head of Mask R-CNN [3] has more than 2.3M parameters and takes $\sim 2.5\times$ computational time (11.4 ms per 100 instances).

3.3. Design Choices of the Mask Branch

We further investigate the impact of the mask branch. We first change C_{mask}, which is the number of channels of the mask branch's output feature maps (*i.e.*, $\mathbf{F}_{mask}$). As shown in Table 2, as long as C_{mask} is in a reasonable range (*i.e.*, from 2 to 16), the performance keeps almost the same. $C_{mask} = 8$ is optimal and thus we use $C_{mask} = 8$ in all other experiments by default.

As mentioned before, before taken as the input of the mask heads, the mask branch's output $\mathbf{F}_{mask}$ is concatenated with a map of relative coordinates, which provides a strong cue for the mask prediction. As shown in Table 3 (2nd row), the performance drops significantly if the relative coordinates are removed (35.7% vs. 31.4%). The significant performance drop implies that the generated filters not only encode the appearance cues but also encode the shape of the target instance. It can also be evidenced by the experiment only using the relative coordinates. As shown in Table 3 (2rd row), only using the relative coordinates can also obtain decent performance (31.3% in mask AP). We would like to highlight that unlike Mask R-CNN, which encodes the shape of the target instance by a bounding-box, CondInst implicitly encodes the shape into the generated filters, which can easily represent any shapes including irregular ones and thus is much more flexible. We also experiment with the absolute coordinates, but it cannot largely boost the performance as shown in Table 3 (32.0%). This suggests that the generated filters mainly carry local cues such as shapes. It is preferred to mainly rely on the local cues because we hope that CondInst is translation invariant.

3.4. How Important to Upsample Mask Predictions?

As mentioned before, the original mask prediction is upsampled and the upsampling is of great importance to the final performance. We confirm this in the experiment. As shown in Table 4, without using the upsampling (1st row in the table), in this case CondInst can produce the mask prediction with $\frac{1}{8}$ of the input image resolution, which merely achieves 34.4% in mask AP because most of the details (*e.g.*, the boundary) are lost. If the mask prediction is upsampled by factor $= 2$, the performance can be significantly improved by 1.4% in mask AP (from 34.4% to 35.8%). In particular, the improvement on small objects is large (from 15.1% to 17.0), which suggests that the upsampling can greatly retain the details of objects. Increasing the upsampling factor to 4 slightly worsens the performance (from

w/ abs. coord.	w/ rel. coord.	w/ $\mathbf{F}_{mask}$	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_1	AR_{10}	AR_{100}
		✓	31.4	53.5	32.1	15.6	34.4	44.7	28.4	44.1	46.2
	✓		31.3	54.9	31.8	16.0	34.2	43.6	27.1	43.3	45.7
✓		✓	32.0	53.3	32.9	14.7	34.2	46.8	28.7	44.7	46.8
	✓	✓	**35.7**	**56.3**	**37.8**	**17.1**	**39.1**	**50.2**	**30.4**	**48.8**	**51.5**

Table 3: Ablation study of the input to the mask head on MS-COCO `val2017` split. As shown in the table, without the relative coordinates, the performance drops significantly from 35.7% to 31.4% in mask AP. Using the absolute coordinates cannot improve the performance remarkably (only 32.0%), which implies that the generated filters mainly encode the local cues (*e.g.*, shapes). Moreover, if the mask head only takes as input the relative coordinates (*i.e.*, no appearance features in this case), CondInst also achieves modest performance (31.3%).

factor	resolution	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	$1/8$	34.4	55.4	36.2	15.1	38.4	50.8
2	$1/4$	**35.8**	**56.4**	**38.0**	17.0	**39.3**	**51.1**
4	$1/2$	35.7	56.3	37.8	**17.1**	39.1	50.2

Table 4: The instance segmentation results on MS-COCO `val2017` split by changing the factor used to upsample the mask predictions. "resolution" denotes the resolution ratio of the mask prediction to the input image. As shown in the table, if without the upsampling (*i.e.*, factor $= 1$), the performance drops significantly (from 35.8% to 34.4% in mask AP). Almost the same results are obtained with ratio 2 or 4.

NMS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
box	**35.7**	56.3	**37.8**	17.1	39.1	50.2
mask	**35.7**	**56.7**	37.7	**17.2**	**39.2**	**50.5**

Table 5: Instance segmentation results with different NMS algorithms. As shown in the table, mask-based NMS can obtain the same overall performance as box-based NMS, which suggests that CondInst can totally eliminate the bounding-box detection. Note that it is impossible for ROI-based methods such as Mask R-CNN to remove bounding-box detection.

35.8% to 35.7% in mask AP), probably due to the relatively low-quality annotations of MS-COCO. We use factor $= 4$ in all other models as it has the potential to produce high-resolution instance masks.

3.5. CondInst without Bounding-box Detection

Although we still keep the bounding-box detection branch in CondInst, it is conceptually feasible to totally eliminate it if we make use of the NMS using no bounding-boxes. In this case, all the foreground samples (determined by the classification head) will be used to compute instance masks, and the duplicated masks will be removed by mask-based NMS. As shown in Table 5, with the mask-based NMS, the same overall performance can be obtained as box-based NMS (35.7% vs. 35.7% in mask AP).

3.6. Comparisons with State-of-the-art Methods

We compare CondInst against previous state-of-the-art methods on MS-COCO `test-dev` split. As shown in Table 6, with $1\times$ learning rate schedule (*i.e.*, $90K$ iterations), CondInst outperforms the original Mask R-CNN by 0.8% (35.4% vs. 34.6%). CondInst also achieves a much faster speed than the original Mask R-CNN (49ms vs. 65ms per image on a single V100 GPU). To our knowledge, it is the first time that a new and simpler instance segmentation method, without any bells and whistles outperforms Mask R-CNN both in accuracy and speed. CondInst also obtains better performance (35.9% vs. 35.5%) and on-par speed (49ms vs 49ms) than the well-engineered Mask R-CNN in `Detectron2` (*i.e.*, Mask R-CNN* in Table 6). Furthermore, with a longer training schedule (*e.g.*, $3\times$) or a stronger backbone (*e.g.*, ResNet-101), a consistent improvement is achieved as well (37.8% vs. 37.5% with ResNet-50 $3\times$ and 39.1% vs. 38.8% with ResNet-101 $3\times$), which suggests CondInst is inherently superior to Mask R-CNN. Moreover, as shown in Table 6, with the auxiliary semantic segmentation task, the performance can be boosted from 37.8% to 38.8% (ResNet-50) or from 39.1% to 40.1% (ResNet-101), without increasing the inference time. For fair comparisons, all the inference time here is measured by ourselves on the same hardware with the official codes.

We also compare CondInst with the recently-proposed instance segmentation methods. Only with half training iterations, CondInst surpasses TensorMask [13] by a large margin (38.8% vs. 35.4% for ResNet-50 and 39.1% vs. 37.1% for ResNet-101). CondInst is also $\sim 8\times$ faster than TensorMask (49ms vs 380ms per image on the same GPU) with similar performance (37.8% vs. 37.1%). Moreover, CondInst outperforms YOLACT-700 [2] by a large margin with the same backbone ResNet-101 (40.1% vs. 31.2% and

method	backbone	aug.	sched.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [3]	R-50-FPN		$1\times$	34.6	**56.5**	36.6	15.4	36.3	**49.7**
CondInst	R-50-FPN		$1\times$	**35.4**	56.4	**37.6**	**18.4**	**37.9**	46.9
Mask R-CNN*	R-50-FPN	✓	$1\times$	35.5	57.0	37.8	19.5	37.6	46.0
Mask R-CNN*	R-50-FPN	✓	$3\times$	37.5	59.3	40.2	21.1	39.6	48.3
TensorMask [13]	R-50-FPN	✓	$6\times$	35.4	57.2	37.3	16.3	36.8	49.3
CondInst	R-50-FPN	✓	$1\times$	35.9	56.9	38.3	19.1	38.6	46.8
CondInst	R-50-FPN	✓	$3\times$	37.8	59.1	40.5	21.0	40.3	48.7
CondInst w/ sem.	R-50-FPN	✓	$3\times$	38.8	60.4	41.5	21.1	41.1	51.0
Mask R-CNN	R-101-FPN	✓	$6\times$	38.3	61.2	40.8	18.2	40.6	**54.1**
Mask R-CNN*	R-101-FPN	✓	$3\times$	38.8	60.9	41.9	21.8	41.4	50.5
YOLACT-700 [2]	R-101-FPN	✓	$4.5\times$	31.2	50.6	32.8	12.1	33.3	47.1
TensorMask	R-101-FPN	✓	$6\times$	37.1	59.3	39.4	17.4	39.1	51.6
CondInst	R-101-FPN	✓	$3\times$	39.1	60.9	42.0	21.5	41.7	50.9
CondInst w/ sem.	R-101-FPN	✓	$3\times$	**40.1**	**62.1**	**43.1**	**21.8**	**42.7**	52.6

Table 6: Comparisons with state-of-the-art methods on MS-COCO `test-dev`. "Mask R-CNN" is the original Mask R-CNN [3] and "Mask R-CNN*" is the improved Mask R-CNN in `Detectron2` [35]. "aug.": using multi-scale data augmentation during training. "sched.": the used learning rate schedule. "$1\times$" means that the models are trained with $90K$ iterations, "$2\times$" is $180K$ iterations and so on. The learning rate is changed as in [36]. 'w/ sem": using the auxiliary semantic segmentation task.

both with the auxiliary semantic segmentation task). Moreover, as shown in Fig. 2, compared with YOLACT-700 and Mask R-CNN, CondInst can preserve more details and produce higher-quality instance segmentation results. More qualitative results are shown in Fig. 4.

4. Conclusions

We have proposed a new and simpler instance segmentation framework, named CondInst. Unlike previous method such as Mask R-CNN, which employs the mask head with fixed weights, CondInst conditions the mask head on instances and dynamically generates the filters of the mask head. This not only reduces the parameters and computational complexity of the mask head, but also eliminates the ROI operations, resulting in a faster and simpler instance segmentation framework. To our knowledge, CondInst is the first framework that can outperform Mask R-CNN both in accuracy and speed, without longer training schedules needed. We believe that CondInst can be a new strong alternative to Mask R-CNN for instance segmentation.

References

[1] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[2] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: real-time instance segmentation," in *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 9157–9166, 2019.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 2961–2969, 2017.

[4] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Advances in Neural Inf. Process. Syst.*, pp. 8024–8035, 2019.

[5] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *USENIX Symp. Operating Systems Design & Implementation (OSDI)*, pp. 265–283, 2016.

[6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 3431–3440, 2015.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2017.

[8] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 9627–9636, 2019.

[9] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2016.

[10] L. Boominathan, S. Kruthiventi, and R. V. Babu, "Crowdnet: A deep convolutional network for dense crowd counting," in *Proc. ACM Int. Conf. Multimedia*, pp. 640–644, ACM, 2016.

[11] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," in *Proc. Advances in Neural Inf. Process. Syst.*, pp. 667–675, 2016.

[12] B. Yang, G. Bender, Q. V. Le, and J. Ngiam, "Condconv: Conditionally parameterized convolutions for efficient in-

Figure 4. More qualitative results of CondInst. Best viewed on screen.

ference," in *Proc. Advances in Neural Inf. Process. Syst.*, pp. 1305–1316, 2019.

[13] X. Chen, R. Girshick, K. He, and P. Dollár, "Tensormask: A foundation for dense object segmentation," in *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 2061–2069, 2019.

[14] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, *et al.*, "Hybrid task cascade for instance segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 4974–4983, 2019.

[15] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 8759–8768, 2018.

[16] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring r-cnn," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 6409–6418, 2019.

[17] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks," in *Proc. Eur. Conf. Comp. Vis.*, pp. 534–549, Springer, 2016.

[18] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 8837–8845, 2019.

[19] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *Proc. Advances in Neural Inf. Process. Syst.*, pp. 2277–2287, 2017.

[20] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy, "Semantic instance segmentation via deep metric learning," *arXiv: Comp. Res. Repository*, 2017.

[21] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan, "Blendmask: Top-down meets bottom-up for instance segmentation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020.

[22] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: Segmenting objects by locations," *arXiv: Comp. Res. Repository*, 2019.

[23] E. Xie, P. Sun, X. Song, W. Wang, D. Liang, C. Shen, and P. Luo, "PolarMask: Single shot instance segmentation with polar representation," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020.

[24] K. Sofiiuk, O. Barinova, and A. Konushin, "Adaptis: Adaptive instance selection network," in *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 7355–7363, 2019.

[25] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proc. AAAI Conf. Artificial Intell.*, 2018.

[26] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 2117–2125, 2017.

[27] 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. Comp. Vis.*, pp. 740–755, Springer, 2014.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Advances in Neural Inf. Process. Syst.*, pp. 91–99, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015.

[30] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv: Comp. Res. Repository*, 2019.

[31] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proc. Int. Conf. 3D Vision (3DV)*, pp. 565–571, IEEE, 2016.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 2980–2988, 2017.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 248–255, Ieee, 2009.

[35] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019.

[36] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," in *Proc. IEEE Int. Conf. Comp. Vis.*, pp. 4918–4927, 2019.