DATNet: Dense Auxiliary Tasks for Object Detection

Alex Levinshtein Alborz Rezazadeh Sereshkeh Konstantinos G. Derpanis
Samsung AI Centre Toronto
{alex.lev, a.sereshkeh, k.derpanis}@samsung.com

Abstract

Beginning with R-CNN, there has been a rapid advancement in two-stage object detection approaches. While two-stage approaches remain the state-of-the-art in object detection, anchor-free single-stage methods have been gaining momentum. We believe that the strength of the former is in their region of interest (ROI) pooling stage, while the latter simplifies the learning problem by converting object detection into dense per-pixel prediction tasks. In this paper, we propose to combine the strengths of each approach in a new architecture. In particular, we first define several auxiliary tasks related to object detection and generate dense per-pixel predictions using a shared feature extraction backbone. As a consequence of this architecture, the shared backbone is trained using both the standard object detection losses and these per-pixel ones. Moreover, by combining the features from dense predictions with those from the backbone, we realize a more discriminative representation for subsequent downstream processing. In addition, we feed the fused features into a novel multi-scale ROI pooling layer, followed by per-ROI predictions. We refer to our architecture as the Dense Auxiliary Tasks Network (DATNet). We present an extensive set of evaluations of our method on the Pascal VOC and COCO datasets and show considerable accuracy improvements over comparable baselines.

1. Introduction

Due to its wide variety of applications, object detection is one of the most active research areas in computer vision. Currently, the top performing methods on the standard benchmarks are two-stage approaches, e.g., [6, 10, 2]. These methods first extract features from the input image. Next, given a set of bounding box proposals, the features are pooled and used for bounding box refinement, classification, and other prediction tasks, such as instance mask prediction [10]. The pooling stage allows the network to independently focus its attention on the relevant features of each proposal, thus simplifying the subsequent prediction tasks, e.g., classification. A major drawback is that the pooling and processing of hundreds of proposals comes with a computational toll. Moreover, two-stage architectures are complex in nature and tend to be difficult to optimize. As a result, a parallel direction has emerged that explores anchor-free, single-stage methods, e.g., [36, 35]. These methods formulate object detection as a set of dense per-pixel prediction tasks, yielding simpler and faster architectures.

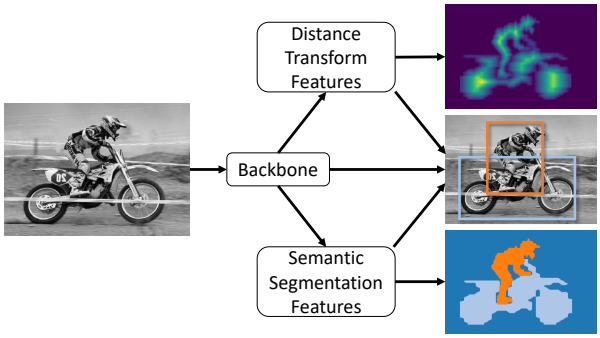

Figure 1. Overview of proposed method. Dense per-pixel distance transform (top) and semantic segmentation (bottom) related features are combined with the initial backbone features to improve the downstream object detection task (right-middle).

Judging by their competitive performance, we believe that anchor-free methods provide a powerful feature substrate for object detection. In this light, we propose a two-stage detector (based on Mask R-CNN [10]) guided by a set of dense per-pixel auxiliary tasks. We refer to our architecture as the Dense Auxiliary Tasks Network (DATNet). While a variety of per-pixels tasks can be designed in support of object detection, in this paper we consider two such auxiliary tasks: semantic segmentation and distance transform (DT). For each pixel, the former contains the semantic category that the pixel belongs to and the latter contains the distance to the closest instance boundary for all foreground pixels and zero for background pixels. Intuitively, the features from the semantic segmentation map are directly useful for instance classification, while the DT features support tasks requiring instance delineation, such as bounding box regression and instance segmentation. We fuse both sets of features with the backbone features, yield-

ing more discriminative features for object detection. We substitute the Mask R-CNN [10] backbone features with the fused features, while keeping all remaining components largely identical to Mask R-CNN (see Fig. 1). By combining per-region losses from Mask R-CNN and the auxiliary per-image losses for semantic segmentation and DT, the resulting hybrid architecture can be optimized end-to-end. In addition, we propose a multi-scale region of interest (ROI) pooling layer that eliminates the manual assignment of feature layers to specific object scale ranges, as used in Mask R-CNN. This enables the network to better exploit the potential of the fused features, leading to a further boost in accuracy. We present extensive evaluations and demonstrate that our method attains significant improvements across common feature extraction backbones on the standard COCO [19] benchmark. Furthermore, we demonstrate that our method is beneficial even in the absence of accurate instance segmentation ground truth. This is demonstrated on Pascal VOC [5], which has no instance segmentation annotations. We use the ground truth bounding boxes as a coarse instance segmentation proxy.

Contributions. In this paper, we make the following three contributions. First, we introduce a novel architecture, the Dense Auxiliary Tasks Network (DATNet), that combines the strengths of single- and two-stage object detectors. In particular, we combine dense per-pixel auxiliary tasks with those from the feature extraction backbone to realize a more discriminative representation for object detection. Second, we propose a multi-scale pooling mechanism with attention over scales to realize the detections. Finally, we present extensive evaluations of our method on the Pascal VOC and COCO datasets and show considerable accuracy improvements over recent baseline methods.

2. Related work

Object detectors. Most modern object detectors can be categorized as either single- or two-stage detectors. Single-stage detectors [25, 22, 20, 36, 35] cast the detection task as (relatively) dense per-pixel object score predictions. Since these detectors simply propagate the image through an efficient convolutional backbone, they are capable of real-time detection. A drawback of these detectors is that their accuracy is typically below their two-stage counterparts.

Alternatively, two-stage detectors [7, 11, 26, 6, 10, 21, 2] combine an object proposal stage that yields a sparse set of object candidate regions with a classifier operating over pooled features of the proposals. A prime example of these detectors is the R-CNN family [7, 6, 26, 10, 8] which has progressively evolved into end-to-end trainable architectures that detect objects and in parallel generate additional outputs, e.g., instance segmentation [10] and 3D shapes [8].

Here, we propose a two-stage architecture that combines features from dense per-pixel predictions of auxiliary tasks (i.e., semantic segmentation and distance transform) with backbone features to improve downstream region processing. Note that these auxiliary tasks are fused early in the feature processing pipeline to enrich the features for downstream processing, as opposed to being relegated as multi-task outputs realized in parallel (e.g., [10]) or used at the later stage of ROI pooling, e.g., [3].

Auxiliary image tasks. Combining multiple related tasks with a shared network has been shown to improve the overall output task accuracies [16, 33]. He et al. [10] showed that using mask prediction as a parallel auxiliary task to object detection not only provides a richer output but also improves the detector accuracy. Several works [4, 34, 17] showed that rather than having parallel branches, features from related tasks can be fused with detector branch features for improved results. However, with the exception of [34], they merge features only after ROI pooling, thereby limiting the effect of contextual information. In the context of panoptic segmentation [15, 31] (i.e., joint object instance and semantic segmentation), region-related features are combined with semantic segmentation to realize the final segmentation output. In contrast to these detection and segmentation methods that treat multiple tasks separately, fuse features on a per-ROI basis, or fuse their features at a late stage, features from our auxiliary tasks are fused earlier with those from the initial feature extraction phase to enrich the features processed onwards.

Multi-stage object detectors [2, 3] recurrently adjust the proposal bounding boxes. These works are complimentary to our own, as we do not attempt to cascade bounding box proposals. Moreover, [2] does not leverage contextual signals provided by auxiliary tasks, as proposed here.

Bai and Urtasun [1] propose a pipeline for instance segmentation, where the final segmentation output is directly inferred from the distance transform of each image to its corresponding object boundary. In our work, the distance transform and semantic segmentation serve as auxiliary tasks, whose features together with the backbone features are passed for further processing to the region proposal network (RPN) and ROI processing stages.

Most similar to our work is Chen et al. [3], where they leverage semantic segmentation as extra contextual features in a cascaded process. Similar fusion is performed by Zhao et al. [34], though it is not the main contribution of their paper. In contrast, we fuse the features earlier (affecting the region proposal stage as well), make use of additional contextual information (distance transform), and generate our semantic segmentation ground truth from instance segmentation rather than making use of extra annotation data.

Attention. Attention is the process of filtering information to highlight the most informative portion of the signal [27]. Over the last few years, attention has been integrated into a variety of visual tasks, including object recognition

and detection [14, 29]. Generically, attention is realized as a conditional gating mechanism, integrated within the network, applied along the spatial dimension alone (e.g., [32]), the channel dimension alone (e.g., [14]), or both dimensions (e.g., [29]). For object detection, [28] use attention to select an image domain, such as traffic or aerial, while [13] use attention to focus on relevant contextual detections.

Most related to our work is the adaptive feature pooling mechanism proposed in [21]. This mechanism allows an object detection architecture to select useful information from different levels of a feature pyramid network (FPN) [18] by fusing these features using an element-wise max operation. Albeit simple and effective, the max operation could still eliminate useful information. In contrast, we concatenate the feature maps from all scale levels of the pyramid, and use the squeeze-and-excitation mechanism [14] to attend to the most useful channels.

3. Technical approach

In this section, we introduce our model, DATNet, and its implementation details. Our model is built upon the Feature Pyramid Network [18] (Fig. 2a). To assist the object detection and instance segmentation tasks, we define auxiliary dense prediction tasks over the entire image (Fig. 2b). Rather than serving solely as auxiliary output losses, we fuse the features from these tasks with the backbone features. The remaining architecture is largely similar to Mask R-CNN, where the features are used to generate region proposals with an RPN and pooled to predict bounding boxes and masks. In contrast to Mask R-CNN, we employ a novel multi-scale pooling mechanism with attention to realize the final outputs. The following subsections provide the details of our two technical contributions.

3.1. Auxiliary image tasks

A variety of auxiliary tasks can be considered. Intuitively, since our final goal is object delineation and classification, our auxiliary tasks should jointly encode this information. We opt for two auxiliary branches that predict the distance transform and semantic segmentation, resp.

Motivated by Bai and Urtasun [1], our first branch (head) is class agnostic and predicts the distance transform (DT) to object instance boundaries. Instead of regressing the distance transform, we normalize and discretize the distances into $K = 4$ bins. Let $D_i(x, y)$ be the distance transform for object instance i. We define $D(x, y) = \max_i \frac{D_i(x,y)}{\max_{xy} D_i(x,y)}$ to be the normalized distance transform. The target bins correspond to 0: background, 1: $0 < D(x, y) \leq 0.2$, 2: $0.2 < D(x, y) \leq 0.4$, and 3: $D(x, y) > 0.4$.

Our second auxiliary branch aims to encode the object class information, and thus is simply a semantic segmentation head. Note that while our target is semantic segmentation, our method requires only instance segmentation ground truth for training. We convert instance segmentation ground truth into semantic segmentation ground truth with a one-hot encoding for every image pixel by pasting instance masks into the appropriate class channel of a semantic segmentation target. Both the DT and semantic heads share the same architecture (but not the same weights), see Fig. 2b. The architecture is similar to Kirillov et al. [15] and is described in detail in Sec. 3.3.

Since both our image heads have categorical targets, they have a similar loss structure. Here we describe the loss for the DT head, L_{DT}. The semantic head loss, L_{seg}, is identical with the lone difference being the number of target classes. While the natural choice is to use the cross entropy loss, due class imbalance and the dominance of the background class, we instead use a variant of the focal loss [20]. For each pixel i, our loss is:

$$L_i = -\sum_k^K \alpha_k (1 - p_k^i)^2 \log(p_k^i) y_k^i, \tag{1}$$

where y_k^i is an indicator variable for pixel i belonging to class k, p_k^i is the model's softmax output, and α_k is a weighting factor for class k. The weighting factor α is set to 0.2 for background and to 1 for all other classes. Our overall DT loss is given by:

$$L_{DT} = \frac{1}{BWH} \sum_i^{B \times W \times H} L_i, \tag{2}$$

where B, W, and H denote the batch size, image width, and height, respectively. Both L_{DT} and L_{seg} are added as auxiliary losses to the original Mask R-CNN loss, $L_{maskrcnn}$. The Mask R-CNN loss contains RPN terms (i.e., bounding box regression and objectness), bounding box head terms (i.e., regression and classification), and a mask head segmentation term; please see [10] for details. The overall loss is:

$$L = L_{maskrcnn} + w_1 L_{DT} + w_2 L_{seg}. \tag{3}$$

Both weights, w_1 and w_2, are set to 0.25 based on cross-validation.

3.2. Multi-scale attention pooling

In the FPN model [18], each proposal is assigned to a single-scale feature map, depending on the proposal area. In particular, small proposals are assigned to finer-resolution levels, while larger proposals are assigned to coarser-resolutions. This hard selection of scale levels, based on manual tuning, may be suboptimal for the following reasons. First, invoking a hard boundary between similar sized proposals may be suboptimal. Second, the most important features for an ROI may not be strongly correlated to its assigned scale level. Depending on the proposal

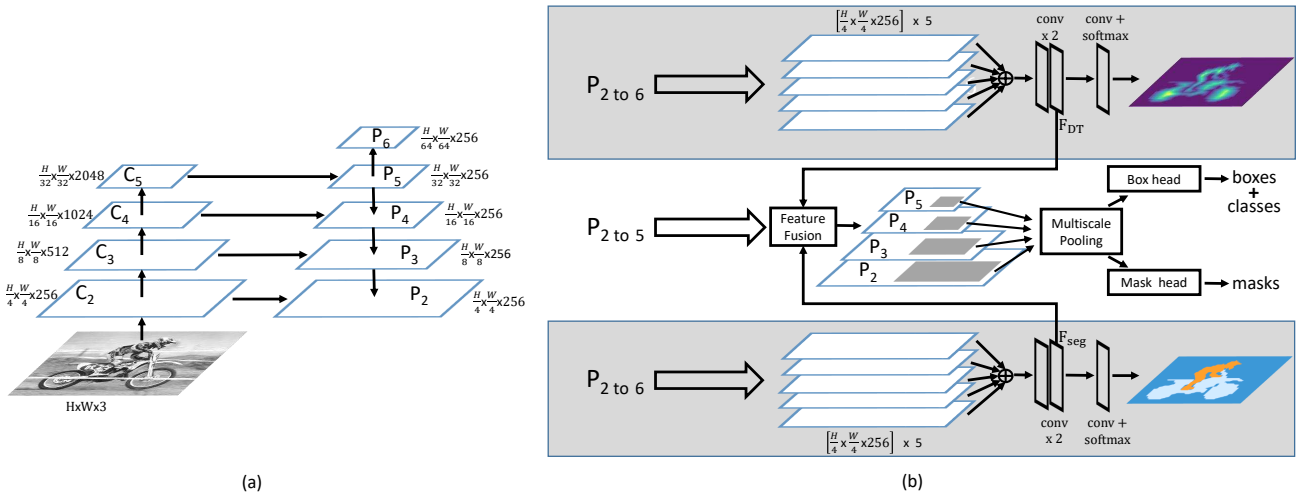

Figure 2. Overview of our proposed architecture. (a) Feature Pyramid Network (FPN) [18] is used as the backbone in our network with an extra layer P_6 generated from subsampling P_5. (b) Features from the auxiliary distance transform (top) and semantic segmentation (bottom) branches are fused with the backbone features, and then pooled using our multi-scale pooling mechanism. The pooled feature are used to predict the class, bounding box, and mask of each proposal.

content, coarser-resolution features may provide important contextual cues, while larger proposals can benefit from the high localization accuracy of finer-resolution features. Finally, for DATNet, where the features maps contain fused features, the FPN's hard selection of single-scale feature maps is not necessarily the best strategy to make use of these fused features.

To address these issues, for each proposal, we concatenate the features from all levels into a single multi-scale feature map and let the network decide the relative importance of each channel via an attention mechanism. We call this process multi-scale pooling with attention (MSPwA). More specifically, first, similar to FPN [18], for each proposal, we create four feature maps, each representing one of the four scale-levels. Next, we concatenate these feature maps (channel-wise) together. In the next layer, we use a squeeze-and-excitation (SE) attention operation [14] to focus further processing on the most informative channels in the concatenated feature map. In particular, the SE attention layer re-weights all the channels by a per channel multiplicative scalar, ranging between zero and one (see Fig. 3).

Following the MSPwA block, the final processing stage consists of a set of ROI heads, i.e., the bounding box and mask predictor heads. In previous work [18, 10], the box predictor head includes two fully connected layers (2fc) to extract features from the ROIs. Since the number of feature channels is now four times larger (due to the concatenation of the scale levels), keeping 2fc would require adding a large number of new parameters to the head and yields a limited performance gain (see Sec. 4.2). Also, the 2fc architecture is optimized for FPNs, and hence, does not necessarily take full advantage of the additional contextual and

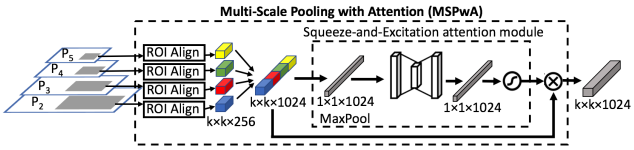

Figure 3. Overview of our multi-scale pooling with attention (MSPwA). Numbers denote spatial resolution and channels.

multi-scale information in the feature map provided by DT and MSPwA. To address these issues, we alter the architecture of the box head by using the ResNet Conv5 (res5) block [12]. The res5 feature extractor has been previously used in [10], but only in their single-scale model. Here, we use res5 to let the network exploit the extra information provided by auxiliary features and multi-scale pooling. We also experimented with other strong heads, such as the one suggested in [20] with four convolutions and one fully connected layer (4conv1fc), but found that res5 returned the highest detection accuracies (see Sec. 4.2).

We propose two different ROI head architectures, referred to as Light and Heavy. Both use the res5 feature extractor for the box head but the latter shares it with the mask head and employs a higher pooling resolution. Additional details are provided in Sec. 3.3.

3.3. Implementation details

Our network is trained end-to-end minimizing the loss in Eq. 3. We implemented our method in PyTorch [24] based on the recent maskrcnn-benchmark code [23].

Auxiliary image tasks. The auxiliary image tasks take as input the four FPN [18] feature maps $(P_2, \ldots, P_5)$ and an extra coarse (P_6) resolution generated by subsampling P_5. All feature maps, excluding P_2, are upsampled to quarter resolution using a sequence of convolution, ReLU, and

$\times 2$ bilinear upsampling operations. The upsampled feature maps are summed together and further processed using a sequence of two convolutions with ReLU. This yields the features F (of size $\frac{H}{4} \times \frac{W}{4} \times 256$), used in feature fusion. For the purpose of auxiliary loss computation, one final convolution and softmax is used to generate per-pixel probabilities. All convolutions are 3×3 with 256 channels, except the final one where the number of output channels corresponds to the number of object classes or distance transform bins.

Feature fusion. For feature fusion, the DT features, F_{DT}, and semantic segmentation features, F_{seg}, are resized using bilinear interpolation to the resolution of each FPN level i and summed with the respective backbone features, yielding new features maps $P'_2, \ldots, P'_5$, where $P'_i = P_i + \text{resize}_i(F_{DT}) + \text{resize}_i(F_{seg})$, and $\text{resize}_i(x)$ resizes x to the resolution of P_i. Section 4.3 evaluates a more elaborate fusion mechanism, which yields additional benefits for proposal generation; however, the simple fusion strategy above, used for all the experiments in this paper, already yields competitive results. The fused features are used by the standard Mask R-CNN RPN, box, and mask heads.

MSPwA. We use ROIAlign [10] to extract four $k \times k \times 256$ feature maps, one from each level of FPN, and concatenate them to form a $k \times k \times 1024$ feature map. We use different k values for the Light and Heavy heads (see the next two paragraphs for details). The channels are then re-weighted using an SE attention layer with max-pooling and a reduction ratio of 16 (see [14] for details).

Light ROI head architecture. The Light architecture uses lower-resolution 7×7 feature maps (MSPwA with $k = 7$) for the box head and higher resolution 14×14 feature maps (MSPwA with $k = 14$) for the mask head. For the box head, we use res5, followed by average pooling. For the mask head, we use the mask head architecture suggested by [10] for FPNs (see Fig. 4 left). In particular, the mask head uses a stack of convolutional layers followed by spatial upsampling to form 28×28 feature maps and another convolutional layer.

Heavy ROI head architecture. Our Heavy architecture is consistent with the single-scale feature extractors proposed in [10]. Feature maps are pooled to $14 \times 14 \times 1024$ ROIs (MSPwA with $k = 14$), followed by a shared res5 feature extractor for box and mask heads. For the mask head, the res5 extractor is followed by spatial upsampling with two convolutional layers yielding 14×14 masks (see Fig. 4 right). In Sec. 4, we show that the higher pooling resolution in the Heavy method leads to higher box AP at the cost of processing time. For masks, there is a trade-off, as the Light method predicts masks at a higher resolution. Since our focus is object detection rather than instance segmentation, improvement of the mask head is left for future work.

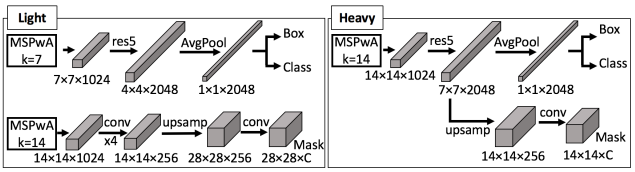

Figure 4. Proposed Light and Heavy ROI head architectures. The numbers denote the spatial resolution and channels of each feature map. All convolutions (conv) are 3×3, except the output conv which is 1×1. 'res5' denotes the fifth stage of ResNet [12] (with a stride of two in its first conv layer), 'upsamp' refers to spatial upsampling with strided transposed convolution, and '$\times 4$' denotes a stack of four consecutive conv layers.

4. Empirical evaluation

We perform extensive ablation studies and compare various instantiations of DATNet (Ours) to other closely related baselines. We make direct comparison with Mask R-CNN [10] and the winner of the 2017 COCO Challenge, PANet [21]. To the best of our knowledge PANet is the leading object detection method that does not employ an ROI cascade. Similar to PANet, we report results with and without multi-scale training (MST). Unlike PANet, we do not make use of synchronized batch normalization.

In the Mask R-CNN appendix [10], end-to-end training details are provided, resulting in higher accuracy; however, in the main manuscript evaluation is based on pre-computed proposals. In our case, end-to-end training is essential, since features are fused prior to RPN. For fairness, we use the end-to-end baselines reported in [23], which are more accurate than [10].

Datasets. We evaluate our method on the standard COCO 2017 [19] and Pascal VOC [5] datasets.

The COCO 2017 [19] consists of 80 object categories with $\sim$118k, 5k, and $\sim$20k images available for training, validation (*val*), and testing on the test server (*test-dev*), respectively. We perform ablation studies on the *val* set and compare to other approaches on the *test-dev* set. We follow the official evaluation metrics [19]: average precision (AP), AP_{50}, AP_{75}, AP_S, AP_M and AP_L, measuring the average precision for different intersection over union (IoU) thresholds (0.5 to 0.95), and object sizes, small (S), medium (M), and large (L). While the main objective of this paper is object detection, since our method returns both bounding boxes and masks, we present results on both.

We use Pascal VOC [5] to show that our approach to integrating auxiliary tasks is beneficial even in the absence of instance segmentation annotations. Following [26], we trained on VOC2007 and VOC2012 trainval (16551 images) and tested on VOC2007 test (4952 images). In place of instance segmentation ground truth, we use the bounding boxes and their corresponding labels as coarse proxies. Since the standard AP_{50} measure used on Pascal VOC is saturated, we report results using the COCO metrics.

Method	Backbone	Object Detection						Instance Segmentation					
		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN [23]	ResNet-50	38.0	59.8	41.4	21.3	40.2	48.1	34.2	56.4	36.1	14.9	36.0	49.5
PANet [21]	ResNet-50	41.2	60.4	44.4	22.7	44.0	54.6	36.6	58.0	39.3	16.3	38.1	53.1
PANet (MST) [21]	ResNet-50	42.5	62.3	46.4	26.3	47.0	52.3	38.2	60.2	41.4	**19.1**	**41.1**	52.6
Ours Light	ResNet-50	40.7	62.1	44.5	23.1	43.0	52.4	36.5	58.9	38.9	16.3	38.4	52.5
Ours Heavy	ResNet-50	41.9	62.5	45.8	24.1	44.3	53.1	36.5	59.6	38.9	16.9	38.3	51.9
Ours Light (2×)	ResNet-50	41.3	62.5	45.2	23.2	43.4	52.9	37.0	59.4	39.4	16.8	38.7	53.2
Ours Heavy (2×)	ResNet-50	42.5	62.7	46.3	24.4	44.5	54.6	36.8	59.9	39.2	17.3	38.3	52.6
Ours Light (MST, 2×)	ResNet-50	43.4	65.0	47.7	25.6	45.8	55.1	38.7	61.8	41.4	18.4	40.4	**55.1**
Ours Heavy (MST, 2×)	ResNet-50	**44.7**	**65.3**	**49.2**	**26.9**	**47.1**	**56.5**	**38.8**	**62.5**	**41.7**	**19.1**	40.6	54.4
Mask R-CNN [23]	ResNeXt-101	42.6	64.9	46.7	25.3	45.8	53.3	37.8	61.4	40.0	17.6	40.3	54.1
PANet [21]	ResNeXt-101 †	45.0	65.0	48.6	25.4	48.6	59.1	40.0	62.8	43.1	18.8	42.3	57.2
PANet (MST) [21]	ResNeXt-101 †	47.4	67.2	51.8	30.1	**51.7**	60.0	**42.0**	65.1	**45.7**	**22.4**	**44.7**	58.1
Ours Light	ResNeXt-101	45.1	66.5	49.6	26.9	48.1	57.2	39.8	63.3	42.6	19.2	42.1	56.8
Ours Heavy	ResNeXt-101	45.8	66.8	50.3	28.0	48.8	57.9	39.7	63.8	42.6	19.8	42.0	56.0
Ours Light (MST, 2×)	ResNeXt-101	47.4	**68.9**	52.2	29.4	50.0	59.7	41.7	65.7	44.7	21.2	43.9	**58.8**
Ours Heavy (MST, 2×)	ResNeXt-101	**48.0**	68.8	**52.9**	**30.3**	50.8	**60.2**	41.4	**65.8**	44.6	21.8	43.5	57.7

Table 1. Evaluation of box and mask AP on COCO *test-dev*. 2× and MST denote double learning schedule and multi-scale training, respectively. No test time enhancements are used. † denotes PANet uses ResNeXt-101 $64 \times 4d$ while we use $32 \times 8d$.

DT head	Semantic head	AP/AP^M	AP_{50}/AP^M_{50}	AP_{75}/AP^M_{75}	AP_S/AP^M_S	AP_M/AP^M_M	AP_L/AP^M_L
✗	✗	37.7/34.3	59.3/55.9	41.0/36.3	21.8/15.6	40.9/36.8	49.6/51.0
✓	✗	39.0/35.2	60.3/56.9	42.3/37.3	22.5/15.9	41.9/37.7	51.3/52.0
✗	✓	38.6/35.0	60.5/57.3	42.0/37.3	22.5/16.2	41.4/37.3	50.6/51.8
✓	✓	**39.3/35.5**	**61.1/57.8**	**42.6/37.6**	**23.2/16.6**	**42.0/37.8**	**51.8/52.4**

Table 2. Ablation of image heads on COCO 2017 *val*. All results are based on the same pooling and ROI processing as Mask R-CNN, thus the first row corresponds to the Mask R-CNN baseline. These results show the performance gains obtained from adding different combinations of the per-pixel tasks.

Training details. Our network starts from public pre-trained ImageNet weights provided in [23]. We use the settings from [10] for training. For each image, we sample 512 regions-of-interest (ROIs) with a positive-to-negative ratio of one to three. We train using SGD with weight decay of 0.0001 and momentum of 0.9. The default data augmentation strategy is random color jitter, image resizing, and random horizontal flipping. In the case of multi-scale training (MST), image resizing randomly selects a size from a predefined list. The remaining settings are dataset and architecture dependent.

For training on COCO, we use the official 1× schedule [23]. With a ResNet-50 backbone, we use a batch of 16 images, training for 90k iterations, with the learning rate initialized at 0.02 and reduced by 10 at 60k and 80k iterations. Similar to [10], images are scaled while maintaining the aspect ratio. We scale the short image edge to 800 pixels while not allowing the long image edge to exceed 1333. For MST, the shorter image edge is randomly chosen between 400 and 1200, while the longer edge cannot exceed 1400. For the ResNeXt-101 ($32 \times 8d$) backbone [30] we use a batch size of eight images, scaling the learning rate and the schedule proportionally [9]. Following the observations in [10] about the positive effect of longer training, we also evaluate a double training schedule (2×), which is the default training schedule in [21].

For Pascal VOC, we use a batch size of eight images, training for 24k iterations, with the learning rate initialized at 0.01 and reduced by 10 at 18k iterations. The image dimensions are scaled in the 600 to 1000 pixel range. We retrain the Faster R-CNN and Mask R-CNN baselines using the same settings. Since our goal for Pascal VOC is to show a use-case without instance segmentation ground truth rather than obtain state-of-the-art accuracy, we report results only with the ResNet-50 backbone.

4.1. COCO evaluation

We compare our model to the Mask R-CNN and PANet baselines. Table 1 shows the results on the COCO test-dev set. Our best method (Heavy), significantly outperforms the corresponding Mask R-CNN baselines with both backbones on box AP (+6.7 mAP for ResNet-50 and +5.4 mAP for ResNeXt-101) and mask AP (+4.6 mAP for ResNet-50 and +3.9 mAP for ResNeXt-101). Compared to PANet, we realize an improved box AP with and without MST: +2.2 mAP for ResNet-50 and +0.6 mAP for ResNeXt-101 with MST, and +1.3 mAP for ResNet-50 without MST. The latter is only 1.1 mAP behind the baseline from [3] which is currently the top method on the COCO instance segmentation leaderboard (modulo bells and whistles). Moreover, the cascaded ROI approach is conceptually orthogonal to our own, and could be added to further improve upon our results. Such a combination is beyond the scope of the current paper. For mask AP, our method is comparable to PANet (+0.6 mAP for ResNet-50 and −0.3 mAP for ResNeXt-101), despite PANet's use of synchronized batch normaliza-

MSPwA	Box F_{ext}	AP/AP^M	AP_{50}/AP^M_{50}	AP_{75}/AP^M_{75}	AP_S/AP^M_S	AP_M/AP^M_M	AP_L/AP^M_L
✗	2fc	37.7/34.3	59.3/55.9	41.0/36.3	21.8/15.6	40.9/36.8	49.6/51.0
✗	4conv1fc	37.8/33.9	58.1/55.0	41.3/35.8	21.2/15.2	40.9/36.5	50.1/50.7
✗	res5	38.6/34.2	59.1/55.7	42.0/36.2	22.1/15.5	41.4/36.6	50.8/50.5
✓	2fc	38.0/34.8	59.9/56.5	41.1/36.9	21.9/15.6	41.1/37.6	49.8/51.7
✓	4conv1fc	38.0/34.4	58.9/55.9	41.2/36.5	20.9/14.8	40.9/37.2	50.1/51.6
✓	res5	**39.5/35.4**	**60.4/57.1**	**43.1/37.6**	**23.4/16.2**	**42.8/37.8**	**52.0/52.6**

Table 3. Ablation of the multi-scale pooling components on COCO 2017 *val*. For this experiment we use the original Mask R-CNN loss for training ($w_1 = w_2 = 0$ in Eqn. 3), thus the first row corresponds to the Mask R-CNN baseline.

Head	Training	AP/AP^M	AP_{50}/AP^M_{50}	AP_{75}/AP^M_{75}	AP_S/AP^M_S	AP_M/AP^M_M	AP_L/AP^M_L
Light	1x	40.3/36.1	61.6/58.2	44.2/38.6	24.1/16.6	43.5/38.5	53.9/53.9
Light	2x	41.0/36.6	61.7/58.8	44.7/38.8	24.8/17.6	43.7/38.7	54.8/54.8
Light	MST 2x	42.8/**38.1**	64.0/60.5	47.3/40.6	26.7/18.6	45.8/**40.6**	56.6/**56.3**
Heavy	1x	41.4/36.2	61.7/58.7	45.0/38.7	24.8/17.8	44.7/38.6	55.2/53.3
Heavy	2x	42.3/36.6	62.2/59.0	46.3/38.9	25.7/17.7	45.2/38.7	56.6/54.0
Heavy	MST 2x	**43.9**/38.0	**64.3/61.2**	**48.5/40.7**	**27.6/19.4**	**47.1**/40.5	**57.4**/55.0

Table 4. Ablation of training settings and ROI head architectures on COCO 2017 *val*. 2× and MST denote double learning schedule and multi-scale training, respectively.

tion. Figure 5 shows qualitative results of DATNet Heavy ResNeXt-101 on COCO *test-dev*.

4.2. Ablation study

In this section, we analyze the effect of the two auxiliary branches and the components of multi-scale pooling. Similar to previous work (e.g., [3, 21]), we use the ResNet-50 FPN backbone for all ablations. We evaluate on the COCO 2017 validation set and report box AP and mask AP^M using the standard COCO metrics.

Auxiliary image heads. We evaluate the effect of each of the auxiliary heads independently, as well as their combination. For this analysis, pooling and ROI processing is identical to Mask R-CNN. Table 2 summarizes the results of this evaluation. As shown in the table, while each head is beneficial on its own, their combination yields the best result (+1.6 mAP).

Multi-scale attention pooling. We evaluate the contribution of multi-scale pooling with attention (MSPwA) across different box feature extractors (Box F_{ext}). We take Mask R-CNN as our baseline for this experiment, i.e., we set $w_1 = w_2 = 0$ in Eqn. 3. Without multi-scale pooling, the two baseline extractors perform similarly, while res5 yields a $+0.9$ box mAP. With multi-scale pooling, the baseline extractors yield minor improvements in box mAP ($+0.3$ for 2fc and $+0.2$ for 4conv1fc). The largest improvement comes from using res5 (with our Light head), yielding $+1.8$ box mAP. We also note that the combination of MSPwA with res5 provides a larger improvement compared to the sum of the improvements from each component.

Heavier head, multi-scale training, and $2\times$ training schedule. We analyze the effect of the remaining enhancements in Table 4. Longer training yields $+0.7$ ($+1.1$) box mAP improvement for DATNet Light (Heavy). Multi-scale training augmentation yields further $+1.8$ ($+1.6$) box mAP for DATNet Light (Heavy). Heavy outperforms Light by $+1.1$ box mAP, but performs similarly for mask mAP. The latter can be explained by the use of a lower feature resolution for the mask head in the Heavy variant.

4.3. Region proposal evaluation

One of the motivating factors for early feature fusion is to obtain better box proposals from RPN. Table 5 analyzes the average proposal recall for different architectures with a ResNet-50 backbone. We report results for 100 and 1000 proposals. While the box and mask heads have elaborate feature extractors, RPN is extremely shallow. As a result, our default feature fusion strategy from Sec. 3.3, which has no learned parameters, has only a marginal effect on proposal recall. Here, we experiment with more advanced fusion strategies that contain multiple Conv+ReLU layers after the sum. The layers are shared across all FPN scales. The last two rows in Table 5 show that adding these extra layers has a positive effect on proposal recall, yielding $+3.4$ ($+5.5$) AR^{100} and $+1.8$ ($+3.4$) AR^{1000}, for two and four layers, respectively. We leave further exploration of feature fusion strategies for future work.

4.4. Pascal VOC evaluation

Here we make the case for using our model despite the unavailability of instance segmentation ground truth. Table 6 illustrates that using coarse proxy masks created using bounding boxes is beneficial, with Mask R-CNN outperforming Faster R-CNN, and our DATNet outperforming Mask R-CNN.

Model	AR^{100}	AR^{100}_S	AR^{100}_M	AR^{100}_L	AR^{1K}	AR^{1K}_S	AR^{1K}_M	AR^{1K}_L
Mask R-CNN [23]	48.2	32.8	55.1	65.6	59.3	48.3	66.3	68.6
Ours Light (*sum*)	49.4	34.3	56.0	66.3	59.6	48.8	66.1	69.5
Ours Heavy (*sum*)	48.8	34.0	55.2	65.6	59.2	48.7	65.4	68.9
Ours Heavy (*sum2conv*)	51.6	37.0	57.5	69.1	61.1	50.3	66.8	72.0
Ours Heavy (*sum4conv*)	**53.7**	**38.8**	**59.8**	**71.1**	**62.7**	**51.7**	**68.3**	**73.9**

Table 5. RPN proposal recall on COCO 2017 *val*. All models use the ResNet-50 FPN backbone. For variants of our method, the feature fusion operation is indicated in brackets.

Figure 5. Qualitative object detection and instance segmentation results of our method on COCO *test-dev*.

Model	AP	AP_{50}	AP_{75}
Faster R-CNN [23]	46.5	76.5	50.4
Mask R-CNN [23]	47.9	76.6	51.9
Ours Light	**49.2**	**77.3**	**53.4**

Table 6. Results on Pascal VOC 2007 test. All models use the ResNet-50 FPN backbone.

Model	Backbone	Time (msec)	Parameters (millions)
Mask R-CNN [23]	ResNet-50	91	44.4
Ours Light	ResNet-50	220	62.3
Ours Heavy	ResNet-50	391	59.9
Mask R-CNN [23]	ResNeXt-101	219	107.4
Ours Light	ResNeXt-101	414	139.2
Ours Heavy	ResNeXt-101	594	136.8

Table 7. Per image inference speed and model size comparison. All results are based on an NVIDIA Tesla P100 GPU.

4.5. Run-time and memory comparison

In Table 7, we compare the inference run-time and size of different models. For the timing analysis, we use eight NVIDIA Tesla P100 GPUs (with one image per GPU) and report the per image per GPU inference time, averaged across COCO 2017 *val* set images. Our models have a $\sim 30\%$ increase in model parameters compared to the respective Mask R-CNN baseline. The inference time of our models is considerably longer, primarily due to the processing of multiple ROIs with a much heavier box head. The long inference time of the Heavy variant (despite it having fewer parameters than Light), is due to the higher resolution of pooling (14×14 versus 7×7) in the box head.

5. Conclusion

In summary, we proposed an object detection architecture, DATNet, that combines favourable aspects from single- and two-stage detectors. Our architecture consists of a common backbone trained with additional per-pixel auxiliary losses, features derived for these auxiliary tasks are fused with those from the backbone, and multi-scale attention for feature pooling for the box and mask heads. Through an extensive set of empirical evaluations this architecture was shown to provide detection improvements over comparable baseline detectors.

References

[1] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, pages 2858–2866, 2017. 2, 3

[2] Z. Cai and N. Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, pages 6154–6162, 2018. 1, 2

[3] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. Hybrid task cascade for instance segmentation. In *CVPR*, 2019. 2, 6, 7

[4] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, pages 3150–3158, 2016. 2

[5] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010. 2, 5

[6] R. B. Girshick. Fast R-CNN. In *ICCV*, pages 1440–1448, 2015. 1, 2

[7] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014. 2

[8] G. Gkioxari, J. Malik, and J. Johnson. Mesh R-CNN. In *ICCV*, 2019. 2

[9] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 6

[10] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *ICCV*, pages 2980–2988, 2017. 1, 2, 3, 4, 5, 6

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *TPAMI*, 37(9):1904–1916, 2015. 2

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 4, 5

[13] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *CVPR*, pages 3588–3597, 2018. 3

[14] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018. 3, 4, 5

[15] A. Kirillov, R. B. Girshick, K. He, and P. Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. 2, 3

[16] I. Kokkinos. UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, pages 5454–5463, 2017. 2

[17] W. Lee, J. Na, and G. Kim. Multi-task self-supervised object detection via recycling of bounding box annotations. In *CVPR*, pages 4984–4993, 2019. 2

[18] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 936–944, 2017. 3, 4

[19] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, pages 740–755, 2014. 2, 5

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *CVPR*, pages 2980–2988, 2017. 2, 3, 4

[21] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *CVPR*, pages 8759–8768, 2018. 2, 3, 5, 6, 7

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, pages 21–37, 2016. 2

[23] F. Massa and R. Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. 4, 5, 6, 8

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 4

[25] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016. 2

[26] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *TPAMI*, 39(6):1137–1149, 2017. 2, 5

[27] J. K. Tsotsos. *A Computational Perspective on Visual Attention*. MIT Press, 2011. 2

[28] X. Wang, Z. Cai, D. Gao, and N. Vasconcelos. Towards universal object detection by domain attention. In *CVPR*, pages 7289–7298, 2019. 3

[29] S. Woo, J. Park, J. Lee, and I. S. Kweon. CBAM: Convolutional block attention module. In *ECCV*, pages 3–19, 2018. 3

[30] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017. 6

[31] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun. UPSNet: A unified panoptic segmentation network. In *CVPR*, 2019. 2

[32] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015. 3

[33] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, pages 3712–3722, 2018. 2

[34] X. Zhao, S. Liang, and Y. Wei. Pseudo mask augmented object detection. In *CVPR*, pages 4061–4070, 2018. 2

[35] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *CoRR*, abs/1904.07850, 2019. 1, 2

[36] X. Zhou, J. Zhuo, and P. Krähenbühl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019. 1, 2