Abstract

Detecting occluded objects still remains a challenge for state-of-the-art object detectors. The objective of this work is to improve the detection for such objects, and thereby improve the overall performance of a modern object detector.

To this end we make the following four contributions: (1) We propose a simple ‘plugin’ module for the detection head of two-stage object detectors to improve the recall of partially occluded objects. The module predicts a tri-layer of segmentation masks for the target object, the occluder and the occludee, and by doing so is able to better predict the mask of the target object. (2) We propose a scalable pipeline for generating training data for the module by using amodal completion of existing object detection and instance segmentation training datasets to establish occlusion relationships. (3) We also establish a COCO evaluation dataset to measure the recall performance of partially occluded and separated objects. (4) We show that the plugin module inserted into a two-stage detector can boost the performance significantly, by only fine-tuning the detection head, and with additional improvements if the entire architecture is fine-tuned. COCO results are reported for Mask R-CNN with Swin-T or Swin-S backbones, and Cascade Mask R-CNN with a Swin-B backbone.

1 Introduction

Occlusion frequently occurs in images of real scenes and is both a benefit and a problem. It is a benefit, because it reveals depth orderings and so contributes to a 3D perception of the scene [10]. However, it is a problem because despite the continual increase in performance of object detectors over the last decade, detection of occluded objects is still a significant deficiency [15, 31, 38].

In this paper, our objective is to improve the detection for objects under occlusion, and thereby improve the overall performance of the object detector. Specifically, we develop a lightweight ‘plugin’ module, that can be inserted into the detection head of any two-stage object detector, e.g. Mask R-CNN [12], to improve the recall of occluded objects. The module simultaneously infers three segmentation ‘layers’: the mask for the target object; the mask of the occluder (the object in front that occludes the target); and the mask of
Figure 1. Improving occluded object detection and instance segmentation. Left: Occlusion is very common in the 3D world, where one object is in front of another and a portion of the scene disappears behind the non-transparent object that is closer to the viewer. Right: For this example from COCO val, a Swin-T + Mask R-CNN Baseline incorrectly detects and segments the target object (the middle of the three cases). However, if the detection head is replaced with our tri-layer plugin, a correct segmentation mask is obtained. The plugin is tasked with predicting the masks of the target object occluder and occludee, and this leads to a better modelling for occluded detection.

the occludee (the object behind that is occluded by the target) within the same detection box. This tri-layer prediction head forces the detection model to explicitly understand the existence of occlusion relationships, and thereby is better able to predict the target object mask. Additionally, we show that the process can be iterated, using the better prediction of the target mask to adapt the detection box for the next tri-layer prediction.

One challenge for training our proposed plugin module lies in the lack of suitable training data – almost all large-scale detection datasets provide annotations on the visible part of objects, but no occlusion information is available. To this end, we propose a scalable pipeline for automatically discovering occluded objects (and their occluders), by running an amodal completion model on publicly available detection datasets, e.g., COCO. Additionally, to verify the occlusion relationships between objects, i.e., occluder or occludee, we adopt an off-the-shelf monocular depth estimator [24] and determine the occlusion relationship based on their relative depth. With this pipeline, we acquire a reliable large-scale data source for training our tri-layer model. We also establish an evaluation dataset to measure the recall performance for occluded objects, distinguishing between the two cases where the target object is partially occluded but the segmentation mask is connected, or where the target object segmentation mask is separated into distinct regions by the occluder.

To summarise, in this paper, we make the following four contributions: (i) We propose a simple ‘plugin’ module that can be inserted into two-stage object detectors, to improve their performance on occluded objects. (ii) We establish a scalable pipeline to determine occlusion relationships between objects, which is used to train the plugin module on publicly available detection datasets. (iii) We set up an evaluation benchmark with real images to measure the recall performance of partially occluded and separated objects. (iv) We show that the plugin module inserted into two-stage detectors can boost the performance significantly, by only fine-tuning the detection head, and with additional improvements if the entire architecture is fine-tuned. We show detection results for Mask R-CNN with Swin-T, Swin-S backbones, and Cascade Mask R-CNN with a Swin-B backbone.

2 Related Work

Object Detection & Instance Segmentation. Methods for object detection and instance segmentation have made great progress by training deep neural networks on large-scale datasets [8, 16, 18]. Two-stage detectors like Faster R-CNN [26] and Mask R-CNN [12] first train region proposal networks to propose candidate objects. These candidates are then fur-
ther refined in their location with a regression head, and categorised by a classifier head. With the development of stronger transformer-based architectures, replacing the original CNN backbones, there have been further improvements in the detection performance. There has also been a similar development in single-stage detectors but we do not consider those in this work. However, both single and two-stage detectors are still suffering when detecting occluded objects.

Amodal Segmentation & Occlusion-Related Datasets. Amodal segmentation refers to the task of segmenting the object as whole, including the portions that are partially occluded. In the recent literature, various datasets have been collected to study occlusion. However, most of these datasets either focus on certain domains/tasks or do not provide a large number of images for training and evaluating models in terms of occlusion for a large variety of categories.

Layered Scene Understanding. Layered representation was originally proposed by Wang and Adelson to represent a video as a composition of layers with simpler motions. Since then, layered representations have been widely adopted in computer vision, e.g., to decompose videos into layers, to improve scene segmentation by explicitly modelling occlusions, temporal consistency, depth ordering, to estimate optical flow, and for novel view synthesis.

Occlusion Handling. To improve object detection and instance segmentation under occlusion scenarios, novel architectures, e.g., compositional networks, have been proposed. Rather than develop a new architecture, ASN and ORCNN ask modern detectors to output both modal and amodal masks of target objects. Additionally, BCNet exploits two-layer graph neural networks to modern detectors for better instance segmentation under occlusion, inferring both the target object and the surrounding objects, while in this paper, we go one step further, and propose a simple, automatic pipeline for estimating objects’ occlusion order, which enables training of the tri-layer plugin with explicit supervision on the occlusion ordering for objects.

3 Detector Architecture and Application

In this section we describe our lightweight plugin module, and its application within an object detector. The module is designed to improve the detection performance on occluded objects. We start by introducing the standard two-stage detector, and then describe the tri-layer plugin architecture and functionality.

Two-stage detector. Given an image detection dataset, \( D = \{(I_1, y_1), \ldots, (I_n, y_n)\} \), a standard two-stage Mask R-CNN detector can be parametrized as:

\[
y_j = \{ (b_j, c_j, m_j) \}^K = \Phi_{\{CLS+BOX;SEG\}} \circ \Phi_{ALIGN} \circ \Phi_{RPN} \circ \Phi_{ENC}(I_j) \tag{1}
\]

where an input image \( I_j \in \mathbb{R}^{H \times W \times 3} \) with a total of \( K \) objects is sequentially processed by a set of operations: an image encoder, \( \Phi_{ENC}(\cdot) \); a region proposal network, \( \Phi_{RPN}(\cdot) \); a region of interest feature alignment, \( \Phi_{ALIGN}(\cdot) \); after predicting the class and box offset (\( \Phi_{CLS+BOX}(\cdot) \)), a binary mask for each RoI is also predicted, \( \Phi_{SEG}(\cdot) \). As a result, \( y_j = \{ (b_j, c_j, m_j) \}^K \) denotes the box coordinates \( (b_j^k \in \mathbb{R}^4) \), category label \( (c_j^k \in \mathbb{R}^C) \), and modal segmentation mask \( (m_j^k \in [0, 1]^{H \times W}) \) of the object, which has been converted from the spatial resolution of RoI align to the original image.

In the following, we refer to a pair of objects that have an occlusion relationship as occluder (the object that is in front of and thus occludes the other one), and occludee (the
3.1 Architecture

In this section, we introduce the architecture details of our tri-layer plugin for better detecting objects under occlusion. Specifically, in Section 3.1.1, we augment the detection module to simultaneously output both occluder and occludee, along with the target object itself; in Section 3.1.2, we introduce the idea of box adjustment, that facilitates the model to reason about the full object coverage with only partial observation, due to partial occlusion or separation; and in Section 3.1.3, we improve the RoI pooling procedure by weighting the feature map with the inferred object mask, effectively preventing the model from concentrating on the occluder/occludee. Figure 2 illustrates the architecture of the plugin.

3.1.1 Tri-Layer Modelling

Here, we augment the instance segmentation head with a tri-layer module that accepts the RoI aligned feature map as input, and outputs the modal segmentation masks of the occluder and occludee for the target object respectively (denoted as Occluder Mask Head and Occludee Mask Head). Specifically, we pass the RoI aligned feature maps into three different mask prediction heads:

\[
\{\hat{b}_j, \hat{c}_j, \hat{m}_j, \hat{\hat{m}}_{j1}, \hat{\hat{m}}_{j2}\}^K = \Phi_{\{\text{CLS+BOX;SEG}\}}(F_j)
\]

(2)

where \(F_j\) refers to the feature map from RoI align, and \(\hat{\hat{m}}_{j1}, \hat{\hat{m}}_{j2}\) refer to the inferred modal segmentation masks for the object’s occluder and occludee respectively. Note that, an object may not have occluder or occludee, and the predicted mask in this case is all zero.

Figure 2. Architecture and function of the plugin module. There are three functions: (a) The tri-layer mask head (MH\(_i\) shown in detail on the right) predicts the mask of the target object (the infant wiping food on their face), the occluder (the dining table), and the occludee (the chair) within the detection box B\(_i\). The feature embeddings of the occluder/occludee branch are concatenated to the target mask embedding as cues to help better predict the target object mask; (b) As shown on the left, the process of predicting the target mask is iterated (index \(i\)), such that the second iteration is able to adjust the initial box predictions and better detect partially occluded / separated objects; (c) After the first iteration, RoI features are pooled according to the predicted target mask to guide the model to focus more on the partially occluded / separated object itself. The notation used is: “\(I\)” for the input image, and “BH”, “MH”, “B”, “C” refer to bbox head, mask head, bounding box, and classification respectively.

Figure 1. An example of the occlusion information for a target occluded object at the back and being occluded), as illustrated in Figure 1. Note that, the role of each object is often relative, thus it can be both occluder and occludee at the same time, depending on its paired objects. In Section 4, we detail the procedure for acquiring the occlusion information, including their estimated amodal segmentation masks and the occlusion orderings.
As shown in Figure 2 (right), the feature embeddings from occluder/occludee branches are further integrated into the target object segmentation branch ($\hat{m}_j$), providing cues to better infer the modal mask of the target object. To act as a proper layering model, we distinguish the order of occluder and occludee by concatenating the feature embeddings in order. Note that, this is in contrast to the previous approach [15], where the embeddings are simply element-wise added, leading to ambiguous occlusion ordering from the commutative rule.

### 3.1.2 Box Adjustment

After predicting the box and segmentation, we carry out a second iteration, to refine the initial predictions. Intuitively, if the network can detect part of an occluded object in the first iteration, an extra iteration will provide the opportunity to adjust the box accordingly, to include any part of the occluded object missed in the first iteration. For the $k$th instance in image $j$,

$$\{b_j, \hat{c}_j, \hat{m}_j, \hat{m}_{j1}, \hat{m}_{j2}\}_k^j = \Phi_{\{CLS+BOX;SEG\}}(\Phi_{ALIGN}(V_j, \hat{b}_j^k)), \forall k \in [1, K]$$

(3)

where $V_j$ denotes the feature map from the visual encoder. The inferred object box ($\hat{b}_j^k$) from the first iteration is used for RoI align to generate refined boxes ($\hat{b}_j^k$), and then we can do tri-layer modelling in the refined boxes.

### 3.1.3 RoI Feature Re-weighting

To guide the model to focus more on the target objects, rather than the occcluders/occludees that may take up a large proportion of the box, we also introduce RoI feature re-weighting with the inferred object mask from the previous iteration:

$$\{\hat{b}_j, \hat{c}_j, \hat{m}_j, \hat{m}_{j1}, \hat{m}_{j2}\}_k^j = \Phi_{\{CLS+BOX;SEG\}}(\Phi_{ALIGN}(V_j, \hat{b}_j^k) \otimes \hat{m}_j^k), \forall k \in [1, K]$$

(4)

where $\hat{m}_j^k$ denotes the inferred segmentation mask from the previous iteration, $\otimes$ denotes the element-wise product between the RoI aligned feature map and the object segmentation from the previous iteration, and $\hat{b}_j, \hat{c}_j, \hat{m}_j$ refer to the output bounding box, object category and instance mask, respectively.

### 3.1.4 Training

We start from publicly released pre-trained models [1]. The fine-tuning procedure is conducted progressively from “Tri-Layer Modelling” to “BBox Adjustment” and then “RoI Feature Re-weighting”, i.e., proposed modules are gradually added after the previous one converges. For tri-layer modelling, both occluder and occludee mask heads use the same structures as the main mask head, except that they are class-agnostic while the main mask head is not. They are trained with binary cross-entropy loss using the inferred ground truth masks, as will be detailed in Section 4.2. The implementation and training are based on MMDet [5]. Please refer to the supplementary for more training details.

### 4 Data Preparation

To properly train our plugin, ground truth occluder/occludee masks for each object are required, as discussed in Section 3.1.4. In this section, we describe the procedure for determining objects’ occlusion order based on their amodal segmentation and depth ordering.
Figure 3. The process of occlusion reasoning via amodal completion and depth estimation. Left: The original image and its depth map. Right: the amodal completion and depth maps for the two objects: Umbrella and Person. In this case we conclude that the Person occludes the Umbrella since: (i) the amodal umbrella has an overlap with the modal person, but no overlap vice versa; and (ii) the average depth map indicates that the depth of the umbrella is greater than that of the person. In this way, we predict that “The umbrella is occluded by the person”.

In general, acquiring amodal segmentations can be very costly, due to the requirement of annotating the occluded parts of the object. To our knowledge, none of the existing large-scale datasets, e.g. COCO [18], LVIS [11], provide amodal segmentation masks. Here, we start by describing a simple, automatic, thus scalable pipeline to approximate object amodal segmentation masks (Section 4.1); in Section 4.2, we further exploit these amodal segmentations to infer the occlusion relationships between paired objects; and in Section 4.3, we detail two different types of occlusion, namely, separation and partial occlusion, and define an evaluation benchmark for occlusion, based on the COCO2017 val set.

4.1 Amodal Completion

Amodal completion aims to infer the amodal mask for an object, given its modal mask, and we want to do amodal completion on COCO. Here, we adopt a similar approach to that of [39], but re-train the amodal completion model on COCO to reduce the domain gap. In detail, instance masks are selected from the COCO dataset and randomly pasted onto training images in order to create artificial occlusions in the images. The amodal completion model is then trained on these images with artificial occlusions to predict the occluded parts, conditioned on the partial (modal) observation. During inference time, for each object, we slightly dilate its modal mask to reduce the gap between connected COCO masks, and the model can thus infer the object’s mask to its full extent, i.e., amodal mask.

While evaluating for amodal completion on different datasets, as shown in Table 1, our model brings a significant improvement over previous approach on COCO. We refer the readers to supplementary for more evaluation details.

<table>
<thead>
<tr>
<th>Model</th>
<th>Eval Dataset</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>[18]</td>
<td>COCO val</td>
<td>81.35</td>
</tr>
<tr>
<td>Ours</td>
<td>COCO2017 val</td>
<td><strong>81.55</strong></td>
</tr>
</tbody>
</table>

Table 1. Amodal Completion on COCO: The comparison between our model and [18]

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Total Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separated COCO</td>
<td>3522</td>
</tr>
<tr>
<td>Occluded COCO</td>
<td>5550</td>
</tr>
<tr>
<td>Occluder Masks</td>
<td>345169</td>
</tr>
<tr>
<td>Occludee Masks</td>
<td>328561</td>
</tr>
</tbody>
</table>

Table 2. Statistics of our generated training and evaluation datasets.

4.2 Occlusion Reasoning

Once we get the amodal mask for each object, the occlusion ordering between a pair of connected objects is inferred in two stages (most easily understood by first looking at the example in Figure 3): First, we compute the intersection between one object’s amodal segmentation and the other object’s modal segmentation. Specifically, for each pair of connected
objects $A$ and $B$, the intersection of $A$’s amodal mask with $B$’s modal mask can be denoted as $I_A$, and the intersection of $B$’s amodal mask with $A$’s modal is denoted as $I_B$. Then $A$ is likely to be occluded by $B$, if $I_A > I_B$. Second, we verify the results by depth estimation, i.e., the occludee should have a greater depth than the occluder. We adopt an off-the-shelf depth estimator [25]. After conducting inference on all COCO images, we compute the average depth of each object over all pixels. Denoting object $A$’s average depth as $d_A$, $B$’s average depth as $d_B$. If $d_A > d_B$ also holds, the result of occlusion ordering is then verified. In this way, we determine the occlusion relationship between objects with greater confidence. For those cases with inconsistent amodal completion and depth verification, we do not assign any occluder-occludee relationship.

With such occlusion reasoning, for each confirmed object pair we have the pseudo ground truth relative occlusion order and ground truth modal masks, to train the tri-layer module (described in Section 3.1.1). Note that, when there are multiple occluders/occludees for one target object, we merge all their modal masks to form the final occluder/occludee mask.

### 4.3 Occluded COCO & Separated COCO for Evaluation

To monitor progress of object detection and instance segmentation under occlusion, we create a benchmark based on the COCO2017 val dataset. We start by defining two different splits for occluded objects, namely, **Separated COCO** and **Occluded COCO**, consisting of separated and partially occluded objects respectively. Unlike previous work [31] that manually selects occluded objects in COCO, we collect the data splits automatically, based on the occlusion reasoning. Specifically, for each object, we can easily check its ground truth modal mask for connectivity. If the mask is split into pieces, then the object is “separated” and should be in the Separated COCO split, for example, the car behind the horses in Figure 4; otherwise, the object is put into the Occluded COCO split as “partially occluded” object, if it is confirmed to be occluded by some other objects, e.g., the airplane at the back in the last image of Figure 4. In this way, we can easily collect two datasets with different occlusion types. Detailed statistics are given in Table 2.

### 5 Experiments

#### 5.1 Datasets and Implementation Details

**Datasets:** We train all models on the COCO2017 training split, together with the occluder/occludee masks obtained in the way described in Section 4.2. We evaluate the performance to detect occluded objects on Separated COCO and Occluded COCO generated in
Section 4.3, and the overall detection performance on both COCO2017 val and COCO2017 test-dev. As for evaluating generalisability, we also test on other datasets like KINS [23] (7517 images for testing), OVIS [22] and OpenImages [16] (details in supplementary).

**Baselines:** We compare the following state-of-the-art architectures, Swin-T + Mask R-CNN, Swin-S + Mask R-CNN and Swin-B + Cascade Mask R-CNN, with or without our designed plugin. In addition, we also compare our tri-layer plugin with the bi-layer modelling approach (inspired by BCNet [15], set all connected objects to be occluders), and compositional network [38] that is specifically designed to handle object occlusion with instance masks.

**Evaluation Metrics:** In addition to the standard metric for COCO2017 val and COCO2017 test-dev, i.e., mAP, we also calculate the recall on Occluded COCO and Separated COCO, i.e., the number of partially occluded / separated objects that are recalled with a fixed number of RPN proposals. Specifically, we treat a partially occluded / separated object being recalled if and only if there is a detection whose confidence > 0.3 and mask IOU > 0.75 with it. mIOU on KINS is calculated, in accordance with that in [22].

### 5.2 Ablation Study

Here, we conduct thorough experiments to validate the effectiveness of different modules in our plugin. As shown in Table 3, we make the following observations: 1) **Tri-layer modelling:** brings a significant improvement in terms of Recall on Occluded COCO (B1 to B3, +75, when only fine-tuning the mask heads; B1 to C2, +96, when fine-tuning the whole network); 2) **Box adjustment:** gives a significant performance boost for BBox mAP, for example, from B1 to B2, +1.9 for only fine-tuning the head, and from B1 to C1, +2.3 for fine-tuning the network; 3) **RoI feature re-weighting:** further improves Mask mAP, Recall on Occluded COCO and Recall on Separated COCO (+0.3/+10/+21 for only fine-tuning the head and +0.1/+7/+15 for fine-tuning the whole network), which is shown by the models, from B4 to B5, and C3 to C4; 4) **Fine-tuning the whole network:** can generally bring improvement on all evaluation metrics, for example, comparing B2-B5 with C1-C4. Notably, only fine-tuning the head could already contribute the majority of the improvement, validating the effectiveness of our proposed module as a general ‘plugin’, which can be inserted into pre-trained detectors, and give quick performance improvement.

<table>
<thead>
<tr>
<th>Model</th>
<th>Tri-Layer Modelling</th>
<th>BBox Adjustment</th>
<th>RoI Feature Re-weighting</th>
<th>Fine-tuning Whole Network?</th>
<th>Recall Occluded</th>
<th>Recall Separated</th>
<th>BBox mAP</th>
<th>Mask mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>3264(58.81%)</td>
<td>1125(31.94%)</td>
<td>46.0</td>
<td>41.6</td>
</tr>
<tr>
<td>B2</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>3296(59.39%)</td>
<td>1141(32.40%)</td>
<td>47.9</td>
<td>42.2</td>
</tr>
<tr>
<td>B3</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>3339(60.16%)</td>
<td>1157(32.85%)</td>
<td>46.0</td>
<td>41.9</td>
</tr>
<tr>
<td>B4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>3400(61.26%)</td>
<td>1187(33.70%)</td>
<td>48.1</td>
<td>42.5</td>
</tr>
<tr>
<td>B5</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3410(61.44%)</td>
<td>1208(34.30%)</td>
<td>48.2</td>
<td>42.8</td>
</tr>
<tr>
<td>C1</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>3367(60.67%)</td>
<td>1170(33.22%)</td>
<td>48.3</td>
<td>42.5</td>
</tr>
<tr>
<td>C2</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>3360(60.54%)</td>
<td>1159(32.91%)</td>
<td>46.3</td>
<td>42.2</td>
</tr>
<tr>
<td>C3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>3434(61.87%)</td>
<td>1208(34.30%)</td>
<td>48.3</td>
<td>42.9</td>
</tr>
<tr>
<td>C4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3441(62.00%)</td>
<td>1223(34.72%)</td>
<td>48.5</td>
<td>43.0</td>
</tr>
</tbody>
</table>

Table 3. Ablation study for adding our plugin to Swin-T + Mask R-CNN. Note that B1-B2 only mask heads are fine-tuned while B2-B3, B3-B4 only bbox heads and mask heads are fine-tuned.
5.3 Comparison with State-of-the-Art

Comparison on COCO. As shown in Table 4, we inject our plugin into a series of popular strong architectures, i.e. Mask R-CNN / Cascade Mask R-CNN with different Swin Transformers as backbone. In all cases, the plugin can always improve recalls for both Occluded COCO and Separated COCO, as well as detection performance on BBox and Mask mAP, sometimes by over 2.5/1.4 (val) and 2.4/1.4 (test-dev) mAP on box and mask predictions. In particular, when compared with bi-layer modelling, our plugin on Swin-T + Mask R-CNN can recall 126 more objects on Occluded COCO and 76 more on Separated COCO, and boost BBox/Mask mAP by 2.2/1.0 (val) and 2.2/1.1 (test-dev) respectively, which shows the effectiveness of our plugin.

<table>
<thead>
<tr>
<th>Detector</th>
<th>Backbone</th>
<th>Plugin</th>
<th>Recall</th>
<th>Recall</th>
<th>val mAP</th>
<th>test-dev mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Occluded</td>
<td>Separated</td>
<td>BBox</td>
<td>Mask</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>Swin-T [1]</td>
<td>–</td>
<td>3264(58.81%)</td>
<td>1125(31.94%)</td>
<td>46.0</td>
<td>41.6</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>Swin-T</td>
<td>bi-layer</td>
<td>3315(59.73%)</td>
<td>1147(32.57%)</td>
<td>46.3</td>
<td>42.0</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>Swin-T</td>
<td>ours</td>
<td>3441(62.00%)</td>
<td>1223(34.72%)</td>
<td>48.5</td>
<td>43.0</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>Swin-S [1]</td>
<td>–</td>
<td>3393(61.14%)</td>
<td>1186(33.67%)</td>
<td>48.5</td>
<td>43.3</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>Swin-S</td>
<td>ours</td>
<td>3473(62.58%)</td>
<td>1261(35.80%)</td>
<td>50.3</td>
<td>44.2</td>
</tr>
<tr>
<td>Cascade Mask R-CNN</td>
<td>Swin-B [1]</td>
<td>–</td>
<td>3491(62.90%)</td>
<td>1279(36.31%)</td>
<td>51.9</td>
<td>45.0</td>
</tr>
<tr>
<td>Cascade Mask R-CNN</td>
<td>Swin-B</td>
<td>ours*</td>
<td>3532(63.64%)</td>
<td>1299(36.88%)</td>
<td>52.1</td>
<td>45.4</td>
</tr>
</tbody>
</table>

Table 4. Comparison with state-of-the-art on different architectures. The plugin gives a performance boost across all the architectures, even for the strongest detector (Swin-B + Cascade Mask R-CNN). * Only Tri-Layer Modelling is applied as Cascade Mask R-CNN has already used multiple iterations.

Comparison on other benchmarks. Here, we directly evaluate the model on the KINS [23] dataset in terms of mIoU. To handle the problem that KINS classes and COCO classes are different, we make a mapping from COCO classes to KINS classes, as detailed in the supplementary. Note that, these models are only trained on COCO, thus resembling a cross-domain generalisation. Specifically, while evaluating the baseline Swin-T + Mask R-CNN, it only achieves 66.6 mIoU, which is slightly lower than the CompositionalNet instance segmentation work [38] with a 67.2 mIoU. However, with our plugin module, the performance can be largely improved, getting 68.5 mIoU and becoming better than [38] on KINS dataset. We refer the reader to the supplementary for more detailed results on KINS, OVIS, and OpenImages.

Comparison of number of parameters and FLOPs. Table 5 compares the number of parameters and FLOPs for different models with/without our plugin. “Tri-Layer Modelling” is a lightweight plugin, only introducing 9.3%, 13.4%, 13.3% more parameters for Swin-S + Mask R-CNN, Swin-T + Mask R-CNN, and Swin-B + Cascade Mask R-CNN, respectively. “BBox Adjustment” introduces more parameters for each model, where the majority of increase comes from the extra bbox head (14M) in the extra iteration. “RoI Feature Re-weighting” does not introduce any extra parameters since it only re-weights the RoI feature using the inferred segmentation mask from the previous iteration. When only fine-tuning the heads, only a small proportion of parameters need to be adjusted, so the training speed is fast.
Table 5. Comparison of number of parameters and FLOPs. “Tri-Layer Modelling” introduces two extra mask heads, but only increases the number of parameters by a small proportion. The parameter increase brought by “BBox Adjustment” is mainly due to the extra bbox head (14M) in the extra iteration. The increase in FLOPs is approximately proportional to the increase in the number of parameters.

5.4 Qualitative Results

In Figure 5, we show qualitative results for inserting our plugin into Swin-T + Mask R-CNN. As can be seen, the baseline model tends to fail in challenging occlusion cases, either over-segmenting the partially occluded (row 1) or under-segmenting the separated (row 2) objects. While our proposed model has largely improved the detection, for example, disambiguating the teddy bears (row 1), and inferring the two separated pieces of the chair that is heavily occluded by the dog (row 2). See supplementary for more examples.

Figure 5. Qualitative results on COCO. Please see the text for more discussion. More qualitative results are provided in supplementary.

6 Conclusion and Future Work

We have proposed a simple ‘plugin’ module for two-stage object detectors that can improve their performance in detecting objects under challenging occlusions. Additionally, we describe a scalable pipeline for automatically identifying occluded and occluding objects in existing benchmarks, to provide training data and an evaluation dataset. Adding the module to a series of popular strong detectors, Mask R-CNN / Cascade Mask R-CNN with different Swin Transformer backbones, leads to consistent performance improvements.

A possible avenue of future work is to improve detection performance of occluded objects in videos, where multiple views of the objects and temporal cues are potentially available to help disambiguate the occlusions.
Acknowledgements. This research is supported by EPSRC Programme Grant VisualAI EP/T028572/1, a Royal Society Research Professorship RP/R1/191132, and a China Oxford Scholarship. We thank Prannay Kaul, Tengda Han and Gyungin Shin for proof-reading.

References


