Supplementary

A Training Details

Our model is implemented with MMDet [5], an open-source object detection toolbox based on Pytorch. For each fine-tuning process, it takes about 20 epochs to converge. We set the initial learning rate to be 0.000001, which is the learning rate at the end of the official COCO training of these models [1], and drop the learning rate by 10 at epoch 16. Weight decay is set to be 0.05, and batch size is 16, following the official training setting. When only the head is fine-tuned, the training process is much quicker because the number of trained weights is greatly cut down. Approximately the training time will be only half of that for fine-tuning the entire network.

B Amodal Completion

In Section B.1, we provide details about the evaluation of the amodal completion model. After that, in Section B.2, we show more visualisation examples to illustrate the effectiveness of the amodal completion model on COCO val.

B.1 Details of Quantitative Evaluation

In this section, we aim to evaluate the performance of different models for amodal completion on COCO. However, one challenge is that COCO does not provide GT amodal masks. We thus borrow the GT amodal masks from COCOA [41] which is a subset of COCO with manual annotation of the amodal segmentation mask for each object. We transfer the GT amodal masks from COCOA to COCO as follows: first, determine the images that are in common between the two datasets; then for a specific object in COCO, within a common image, determine if COCOA provides an amodal mask by comparing their modal masks using IoU. This is necessary because the modal mask in COCOA might be slightly different from that in COCO. If the IoU is greater than 0.7, then the match is accepted, and the amodal mask is used. As a result, we evaluate on 450 objects in COCO2017 val.

As Table 1 shows (result of [39] on COCOA val is as reported in their paper), the performance of [39] drops significantly when applied to COCO2017 val. In contrast, our model achieves better performance on COCO2017 val than [39] on COCOA val. In Section B.2, we provide qualitative results from both our model and [39]. We conjecture that the substantial performance drop of [39] is due to the domain gap between COCO and COCOA, *e.g.*, the annotations in COCO tend to have a gap between objects, and even for the same object the modal annotation mask in COCO and COCOA might be slightly different, thus the model trained on COCOA will struggle to generalise to COCO val.

B.2 Qualitative Comparison

Figure 6 shows qualitative results of our model and the model of [39] for amodal completion on COCO2017 val. We can observe that our model can generate significantly better amodal segmentation masks, and reason about the occluded parts of the objects.

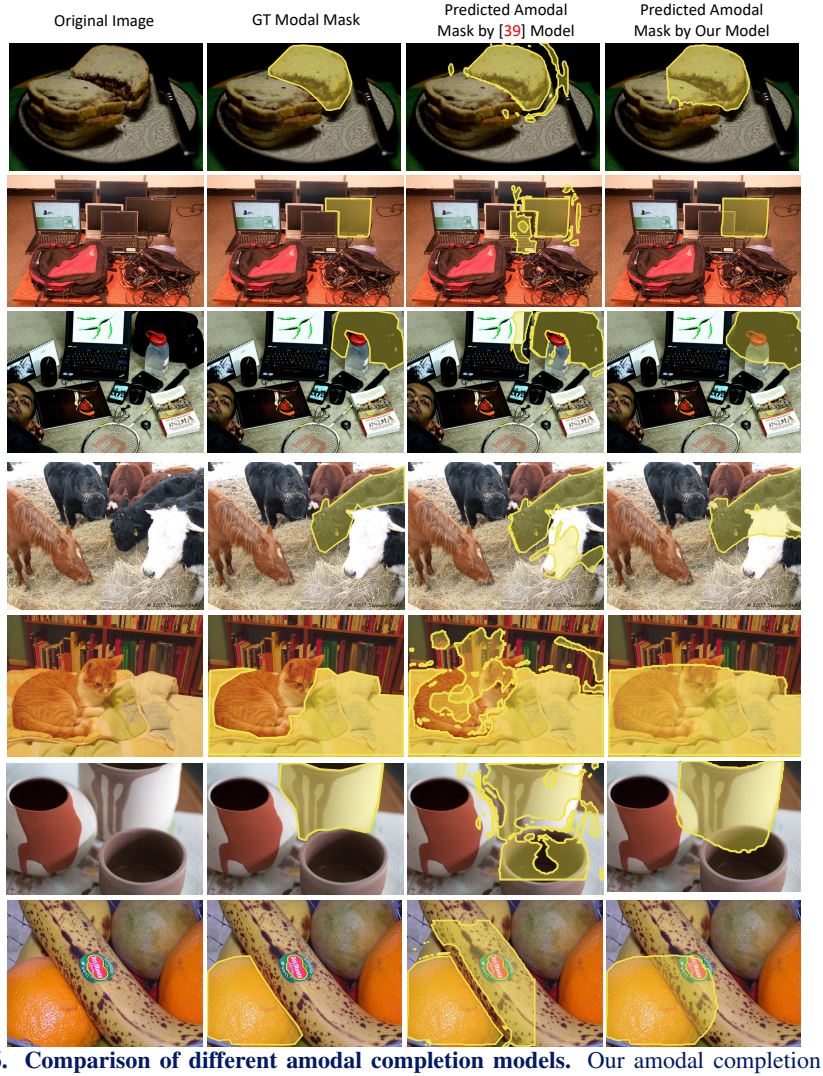

Figure 6. Comparison of different amodal completion models. Our amodal completion model performs significantly better than the model of [39].

C Examples of Generated Training Data

In Figure 7, we show visualisation of the results from our automatic pipeline that can infer occluder and/or occludee masks for target objects in COCO2017 train.

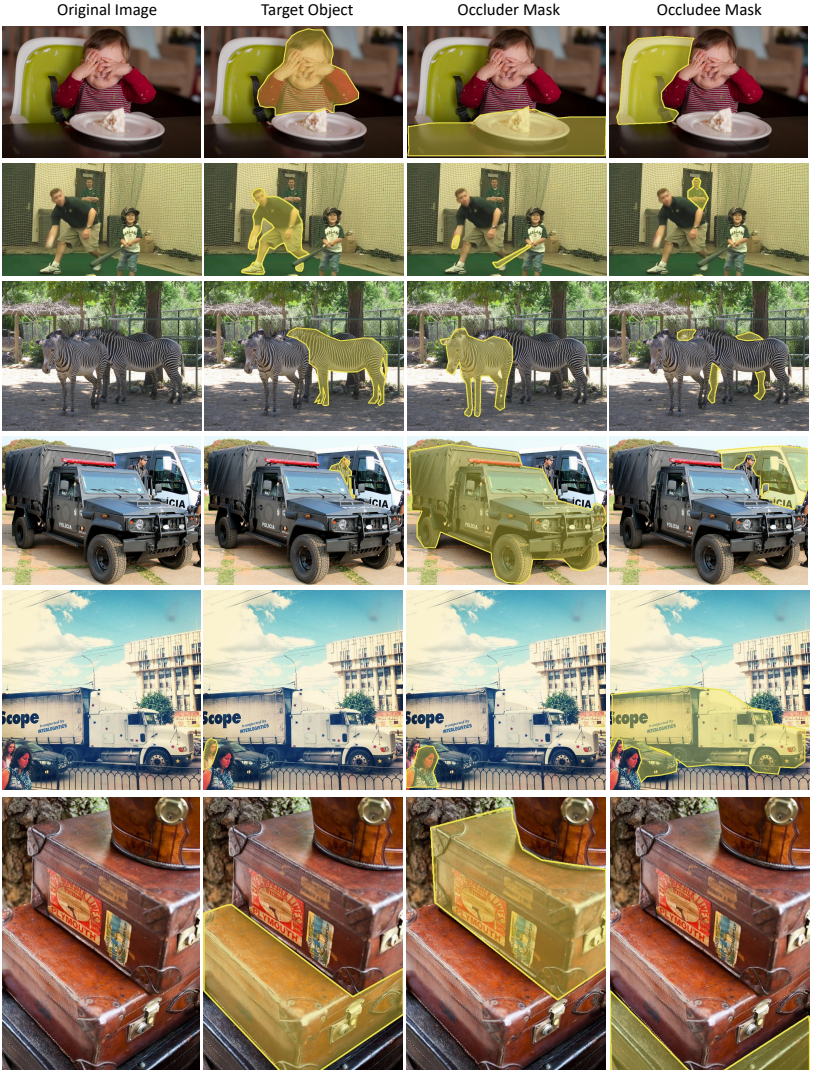

Figure 7. More examples of automatically generated training data. For each row, from left to right is: original image, the object of interest (target object), its occluder mask, and its occludee mask.

D Additional Qualitative Comparison on COCO

Figures 8 and 9 provide more qualitative detection examples to illustrate the effectiveness of our plugin over the baseline (Swin-T + Mask R-CNN) on partially occluded objects and separated objects. In both cases, our plugin can systematically solve two common failure patterns of the baseline detector: (1) Over segmentation, where part of the occluder or surrounding objects is included in the mask; (2) Under segmentation, where only part of the partially occluded / separated object is included in the mask.

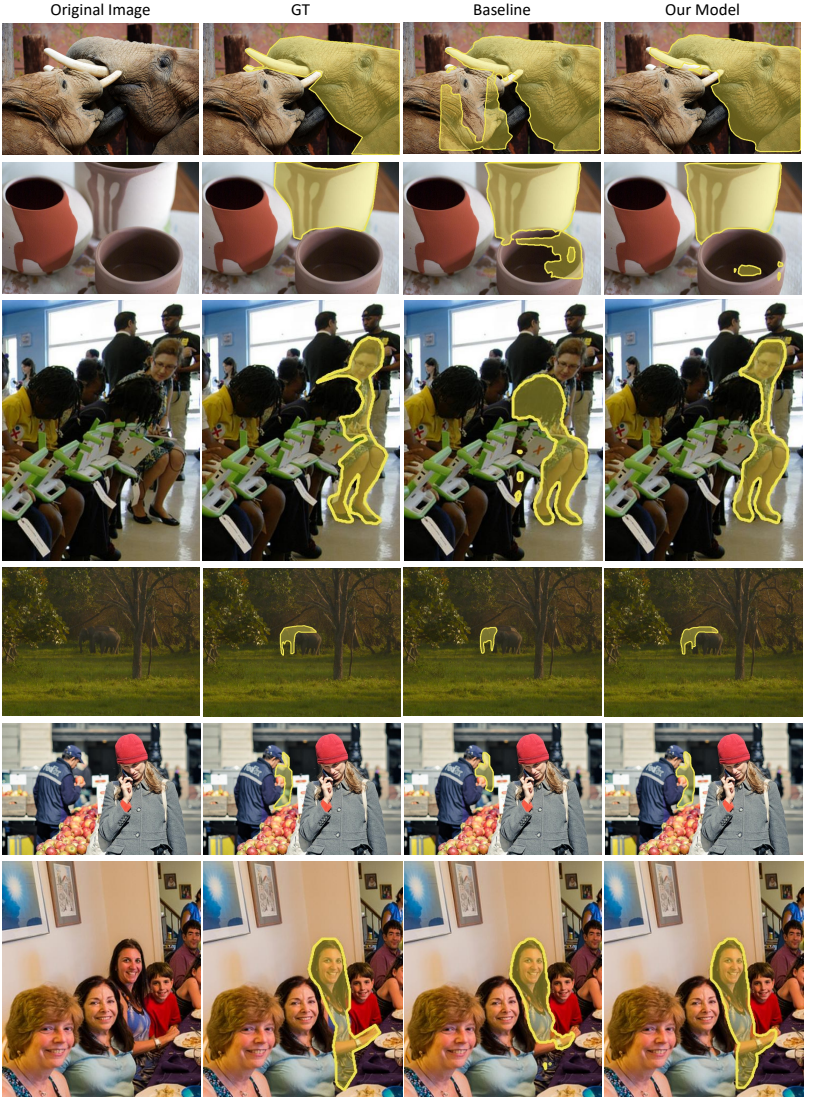

Figure 8. Additional qualitative comparison of our model with baseline on Occluded COCO. Our model can systematically solve the common failure patterns of over-segmentation (the mask is too large, and includes part of occluder) (Row 1-3) and under-segmentation (the mask is too small) (Row 4-6) for partially occluded objects.

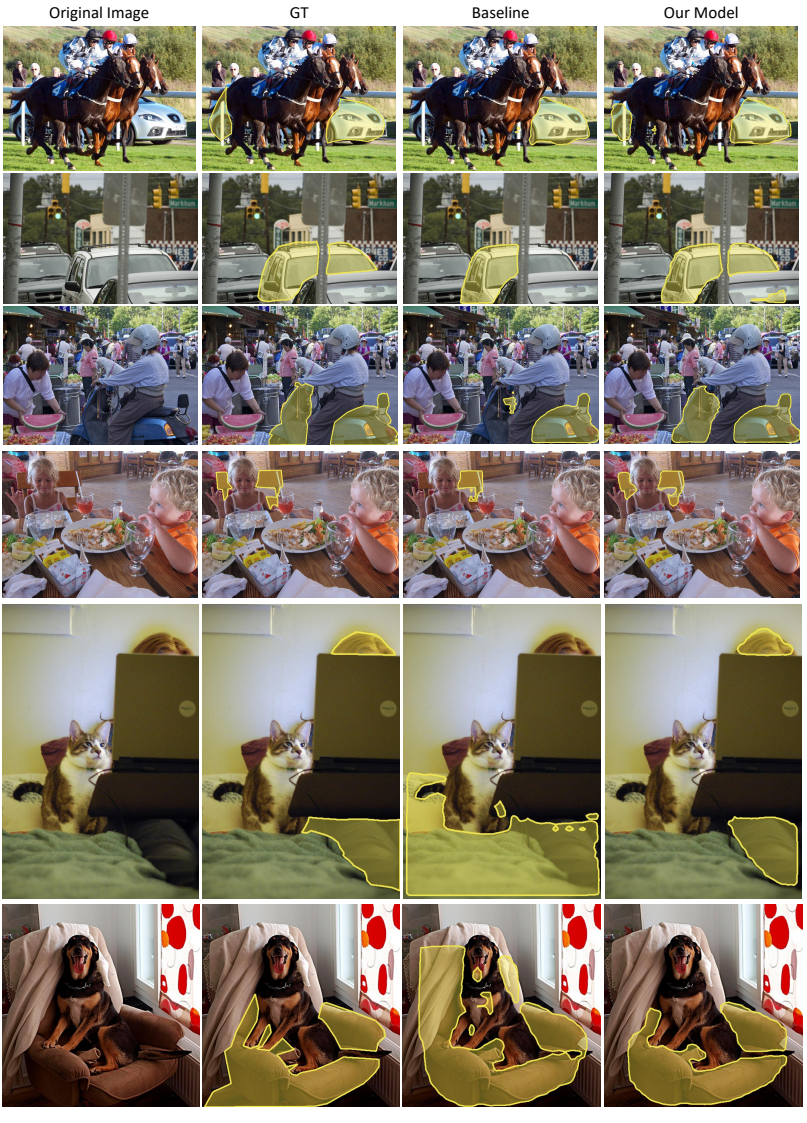

Figure 9. Additional qualitative comparison of our model with baseline on Separated COCO. Our model could systematically solve the failure patterns of under-segmentation (the mask is too small, and only includes part of the separated object) (Row 1-4) and over-segmentation (the mask is too large, and also includes part of the occluder or surrounding objects) (Row 5-6) for separated objects.

E Results on Other Datasets

In this section, we include additional experimental results for comparison between our model and baseline (Swin-T + Mask R-CNN) on OpenImages [16], OVIS [22] and KINS [23], as well as details for evaluation on each dataset. Section E.1, Section E.2 and Section E.3 provide quantitative results and evaluation details for OpenImages, OVIS and KINS, respectively, while qualitative results are shown in Section E.4.

E.1 Quantitative Results on OpenImages

OpenImages is a large-scale image dataset with manual annotations for some object masks together with the labels, indicating whether the object is occluded/truncated/crowd/depiction/inside or not. To evaluate the model's capability to detect partially occluded / separated objects, we collect a subset of objects with masks in OpenImages Test Set, which are labelled as occluded but not truncated/crowd/depiction/inside, and denote the subset as **Only Occluded OpenImages Test**. We further divide these objects into an Occluded set and a Separated set, depending on whether their masks are connected or not, like the division for COCO in the main paper. As a result, there are 3356 objects in total, with 2103 Occluded and 1253 Separated, in 2348 images.

Table 6 shows that our plugin can improve the performance of the baseline model in terms of recall and mIoU on both the Occluded objects and Separated objects. The recall and mIoU for Occluded objects only shows marginal improvement, because the selected "Occluded objects" are relatively easy and already well-detected by the baseline (over 75%). Therefore, the advantage of our plugin on "Occluded objects" is not so significant.

Method	Only Occluded OpenImages Test				Sampled OVIS			
	Recall Occluded	Recall Separated	mIoU Occluded	mIoU Separated	Recall Occluded	Recall Separated	mIoU Occluded	mIoU Separated
Baseline	1551	607	74.0	64.2	3960	1587	**61.1**	49.8
Baseline + Our Plugin	**1569**	**672**	**74.1**	**65.4**	**3994**	**1673**	60.5	**50.3**

Table 6. Results on **Only Occluded OpenImages Test** and **Sampled OVIS**. For both datasets, the plugin slightly improves over the baseline's performance, particularly on Separated objects.

E.2 Quantitative Results on OVIS

OVIS (Occluded Video Instance Segmentation) is a dataset with videos of occluded objects. For each object in the training set, it has mask annotations as well as a manual occlusion label to be 'no occlusion' / 'slight occlusion' / 'severe occlusion'. In order to evaluate on reasonably distinct frames, we pick 1 frame every 10 frames. Then we collect an evaluation image dataset (denoted as **Sampled OVIS**) containing 4443 images where we can calculate each detector's recall of the Occluded objects and Separated objects (the division into 'Occluded' and 'Separated' is the same as in OpenImages). There are in total 7265 Occluded objects and 5187 Separated objects.

From Table 6, we can see that recall on Occluded objects and Separated objects of the baseline can be consistently boosted by the plugin. In terms of mIoU for Occluded/Separated

objects, there is no significant improvement from the plugin. These failure cases are mainly due to motion blur or require temporal context. We leave this to future work.

E.3 Quantitative Results on KINS

As mentioned in Section 5.3, we evaluate on the KINS dataset by directly evaluating the model that has been trained on COCO. To handle the problem that KINS classes and COCO classes are different, we make a mapping from COCO classes as in Table 7. Note that 'misc' is not mapped to any COCO class, and 'misc' objects are not evaluated on.

KINS Class ID	KINS Class Name	Mapped to COCO Class Name
1	cyclist	person
2	pedestrain	person
3	rider	person
4	car	car
5	tram	train
6	truck	truck
7	van	truck
8	misc	none

Table 7. Mapping from KINS classes to COCO classes for evaluation.

Since [38] has not released their code and model, it is difficult to make a fair comparison. In our evaluation, we test the baseline models and our model based on Swin-T + Mask R-CNN on the KINS test set, and calculate the mIoU following [38]. In particular, during inference time, we input the GT box to the models because [38] also inputs the GT amodal box to their model for instance segmentation inference which suits their setting.

Method	mIoU
Yuan *et al* [38]	67.2
Swin-T + Mask R-CNN	66.6
Swin-T + Mask R-CNN + Bi-Layer	67.0
Swin-T + Mask R-CNN + Our Plugin	**68.5**

Table 8. Comparison with [38] on KINS. With our plugin the baseline model can achieve a better performance than [38].

Note that, the performance of our model is under-estimated with this evaluation protocol for the following reasons: (i) KINS classes and COCO classes are different. We use a mapping from KINS classes to COCO classes, but this adds to the difficulty for our model to detect these KINS objects. (ii) KINS annotations and COCO annotations are different, adding to the difficulty of adapting our models to test on KINS. (iii) There could also be a domain shift.

Our model still outperforms the baseline model and the previous approach [38] (shown in Table 8), demonstrating the effectiveness of our plugin module.

E.4 Qualitative Results on Other Datasets

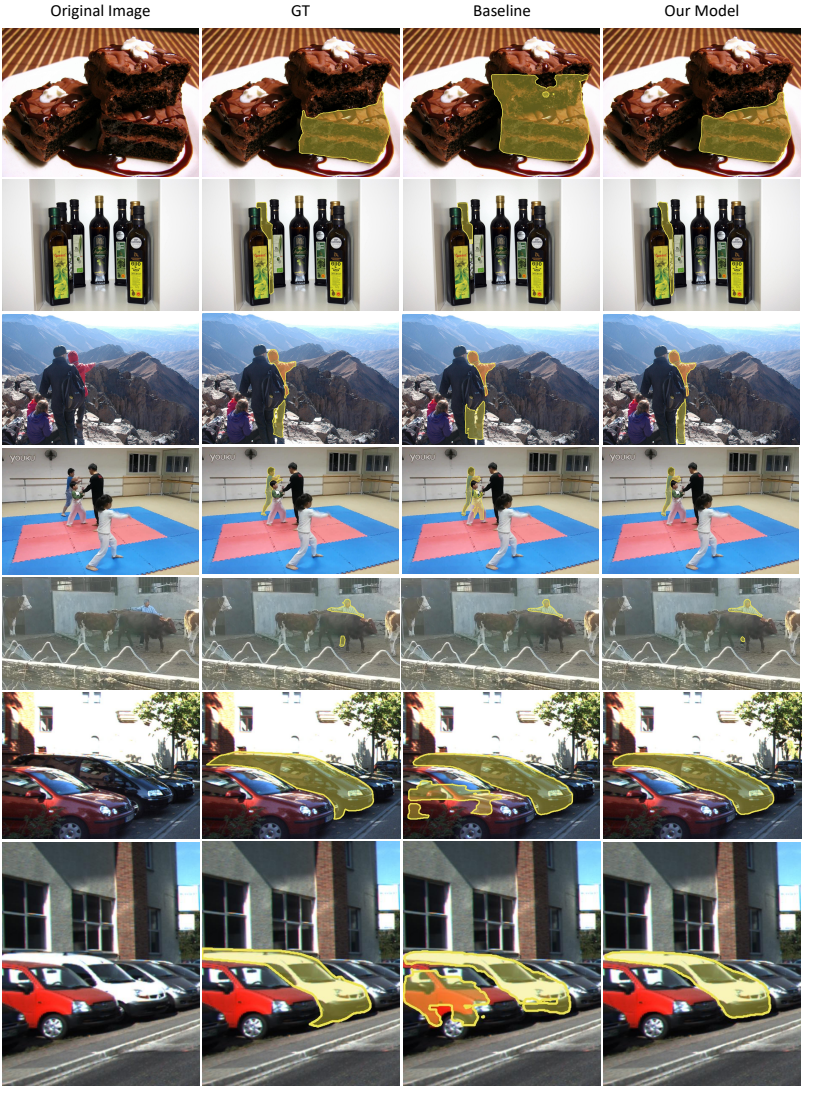

Figure 10. Qualitative comparison on other datasets. Though not trained on these datasets, compared with the baseline, our model can solve the failure patterns of over-segmentation (including part of the occluder in the mask) (Row 1, 3, 4, 6, 7) and under-segmentation (Row 2, 5) for both partially occluded (Row 1, 2, 4, 6, 7) and separated objects (Row 3, 5). OpenImages: Row 1-3; OVIS: Row 4-5; KINS: Row 6-7.

Figure 10 shows examples where our plugin solves the baseline's failure patterns as mentioned in Section D on OpenImage, OVIS and KINS, qualitatively illustrating the effectiveness of our designed plugin when generalised to other datasets.

F Other Discussions

F.1 Number of Iterations

For our current plugin, we apply two iterations of the tri-layer mask heads. We have also experimented with applying the module three times, and the comparison is shown in Table 9.

Number of Iterations	Recall Occluded	Recall Separated	BBox mAP	Mask mAP
Baseline*	3264(58.81%)	1125(31.94%)	46.0	41.6
2	**3434(61.87%)**	**1208(34.30%)**	48.3	42.9
3	3401(61.28%)	1194(33.90%)	**48.7**	**43.0**

Table 9. Comparison of different number of iterations on Swin-T + Mask R-CNN. There is not much difference between 2 and 3 iterations. *Baseline denotes original Swin-T + Mask R-CNN without our plugin.

The mAP performance of three iterations is similar to using it twice, while performance on occluded objects becomes worse. For this reason, we only apply it twice.

F.2 Class-Agnostic v.s. Class-Specific

For both bi-layer and tri-layer modelling, we have two choices of the occluder/occludee heads – either to be class-agnostic or class-specific. Note that bi-layer modelling only has one extra occluder head to predict all surrounding objects of the target object, while tri-layer modelling has a pair of occluder/occludee heads to predict the occluders/occludees of the target object, and can capture the occlusion ordering of different objects.

If an occluder/occludee head is class-agnostic, it only outputs one mask prediction; if the occluder/occludee head is class-specific, it outputs 80 mask predictions and the final result is the i-th mask prediction where i is class prediction of the instance. The results of both bi-layer and tri-layer modelling under class-agnostic and class-specific settings are shown in Table 10.

Bi-Layer/Tri-Layer Modelling	Class-Specific/ Class-Agnostic	Recall Occluded	Recall Separated	BBox mAP	Mask mAP
Baseline*	-	3264(58.81%)	1125(31.94%)	46.0	41.6
Bi-Layer	Class-Specific	3315(59.73%)	1147(32.57%)	46.3	42.0
Tri-Layer	Class-Specific	3358(60.50%)	1166(33.11%)	46.2	42.2
Bi-Layer	Class-Agnostic	3339(60.16%)	1147(32.57%)	46.3	42.2
Tri-Layer	Class-Agnostic	3360(60.54%)	1159(32.91%)	46.3	42.2

Table 10. Comparison of bi-layer and tri-layer modelling under class-specific and class-agnostic settings. In both settings, tri-layer modelling outperforms bi-layer modelling. *Baseline denotes original Swin-T + Mask R-CNN without our plugin.

We can observe that in both class-agnostic and class-specific settings, tri-layer modelling outperforms bi-layer modelling.