Appendix

A Further Illustrations and Pseudo Codes

Here, we describe the implementations of the two designed modules in detail. Particularly, IoU-ESA and DCW are summarized into PyTorch-like code in Algorithm 1 and Algorithm 2, respectively. The complete code is provided in the file of the supplementary material.

For IoU-ESA, it is defined in Eq.(3) in the paper, without newly added parameters. The key difference with the original self attention is that the IoU matrix calculated among proposal boxes, is element-wise multiplied with the attention matrix. Both the IoU and $attn$ are reflecting a kind of similarity of different proposal features.

For the DCW module, two light weights projection heads are added to generate two channel masks for classification and localization, respectively. Because of the the two fc bottlenecks, the increase of parameters and calculations is also small. This module strengthens the two head features for a better utilization of object queries. It can be seen in Fig.1 in the paper.

Algorithm 1 IoU-ESA code (PyTorch-like)

```
# x: input tensor (N, d);
# iou: (N, N)

def IoU-ESA(x, iou):
    q, k, v = fc_qkv(x).chunk(3, dim=-1)
    # nhead: num of heads in attention
    q = q.view(N, nhead, -1).transpose(0, 1)
    k = k.view(N, nhead, -1).transpose(0, 1)
    v = v.view(N, nhead, -1).transpose(0, 1)

    attn = torch.bmm(q, k.transpose(1, 2))

    # Eq (3). in the paper
    attn = torch.exp(attn)
    attn = attn * iou[None, :, :]
    attn = attn / torch.sum(attn, -1, keepdim=True)

    # value routing and output projection
    attn = torch.bmm(attn, v)
    attn= attn.transpose(0, 1).view(N, d)
    attn_out = fc_out(attn)

    return attn_out
```

B Analyze result on CrowdHuman

We analyze self attention and dimension of features in Sec.3.1 in the paper. Results on MS-COCO illustrate that geometry prior is expected to guide self

Algorithm 2 DCW code (PyTorch-like)

```
# obj_queries: (N, d)
# roi_feats: roi features, output of dynamic convs (N, s*s, d)

def DCW(pro_feats, roi_feats):
    # Eq.(4). in the paper
    # mask: (N, d)
    mask_cls = sigmoid(fc2(fc1(obj_queries)))
    mask_reg = sigmoid(fc4(fc3(obj_queries)))

    # roi_feats: (N, s*s, d)
    roi_feats_cls = roi_feats[:, None, :] * mask_cls
    roi_feats_reg = roi_feats[:, None, :] * mask_reg

    # roi_feats: (N, s*s*d)
    roi_feats_cls = roi_feats_cls.flatten(1)
    roi_feats_reg = roi_feats_reg.flatten(1)

    # obj_feats: (N, d)
    obj_feats_cls = fc5(roi_feats_cls)
    obj_feats_reg = fc6(roi_feats_reg)

    return obj_feats_cls, obj_feats_reg
```

attention and disentangle features for the two tasks in detection is effective. Here, we provide the the results on CrowdHuman in Tab. 1 and Tab. 2. Performance on CrowdHuman also illustrate our motivation.

Table 1. Analysis on self attention. Three different models, including origin sparse R-CNN with multi-head self attention (MSA), without self attention and attention matrix replaced by IoU matrix (IoU-MSA), are trained on CrowdHuman. AP and mMR indicate the Average Precision and Log-average Miss Rate, respectively.

Method	AP ↑	mMR ↓	Recall ↑
MSA	88.8	49.9	95.8
w/o MSA	81.5	65.5	95.5
IoU-MSA	86.8	54.2	95.7

C Implementation on DETR and discussions

We also insert the two modules, IoU-ESA and DCW, into DETR [1]. DETR and sparse R-CNN [3] both are of cascade structures, and updating object queries for the next stage. In sparse R-CNN, proposal boxes are inputs of the model, which are parameters at the initial stage and output bounding boxes from last stage at later. IoU matrix is computed among proposal boxes. However, there are no bounding boxes as input in DETR. Therefore, IoU is set to 1 for all elements at the first stage, and in the later stage, it is computed in the same as sparse

Table 2. Analysis on feature disentanglement on CrowdHuman.

Method	AP ↑	mMR ↓	Recall ↑
En. (original)	89.2	48.3	95.9
Dis. (half dim)	88.6	49.7	95.5
Dis. (full dim)	89.1	48.5	95.8

R-CNN. The channel masks m_c and m_r, given by DCW, are generated from the object queries before cross attention, and then perform on object queries after cross attention.

Table 3. Implementation results on DETR. ResNet-50 is utilized as backbone, and the number of object queries is set as 100.

Method	Params	FLOPs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
DETR	41M	86G	34.9	55.5	35.9	14.5	37.6	53.8
DETR + ours	48M	86G	35.4	56.8	36.7	15.4	38.0	54.1

We train it for 50 epochs, and the learning rate is dropped by a factor of 10 at 40th epoch. All other settings are the same as DETR. As shown in Tab. 3, our work reaches 35.4, which slightly gains 0.5 AP from DETR. It also behaves well on the other five metrics. The number of parameters in our work rises from 41M to 48M, and FLOPS keeps the same. Although our work improves the performance on sparse R-CNN significantly, it doesn't do so well on DETR. We notice that quries in DETR obtain all the information from the whole image, which is different from sparse R-CNN. Furthermore, since each query is $1 \times 1 \times C$ without spatial dimension, m_c and m_r can not apply for different spatial positions like SE-Net [2]. This indicates that the proposed modules are targeted to sparse R-CNN, in which RoI-align is performed to get the local object region.

D Visualizations

As our two designed modules enhance the origin sparse R-CNN, the performance increases stably based on different backbones. The convergence speed of the two works are shown in Fig. 1. Note that 1× and 2× (12 and 24 epochs) training schedules are also plotted to indicate the convergence rate. Compared with sparse R-CNN, our work achieves a better AP from the beginning of training, our work finally gains 1.6 AP at 36th epoch. Some visual results on COCO are giving in Fig. 2, which organizes the three columns in the sequence of sparse R-CNN, ours and ground truths. In simple scenarios, there are no big differences between sparse R-CNN and our work. At the first row, both sparse R-CNN and our work have done a good job. The rest four rows' scenarios are more complex

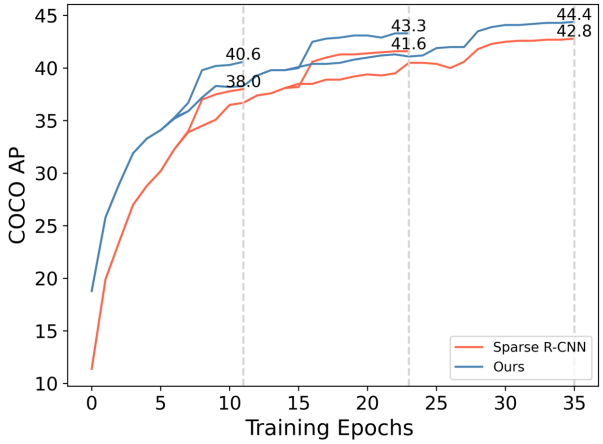


Fig. 1. Convergence curves of sparse R-CNN and ours on COCO val2017. Both models have the same training settings with ResNet-50 and 100 proposals. Our work gains higher AP than sparse R-CNN from the beginning to the end.

than the first one, we can see that the detection result of our work is better than sparse R-CNN. The targets in complicated scene are detected by our work. Similar phenomena can be observed in Fig. 3 and Fig. 4. Detection results on CrowdHuman are also shown in Fig. 5.

References

1. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
2. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
3. Sun, P., Zhang, R., Jiang, Y., Kong, T., Xu, C., Zhan, W., Tomizuka, M., Li, L., Yuan, Z., Wang, C., et al.: Sparse r-cnn: End-to-end object detection with learnable proposals. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14454–14463 (2021)

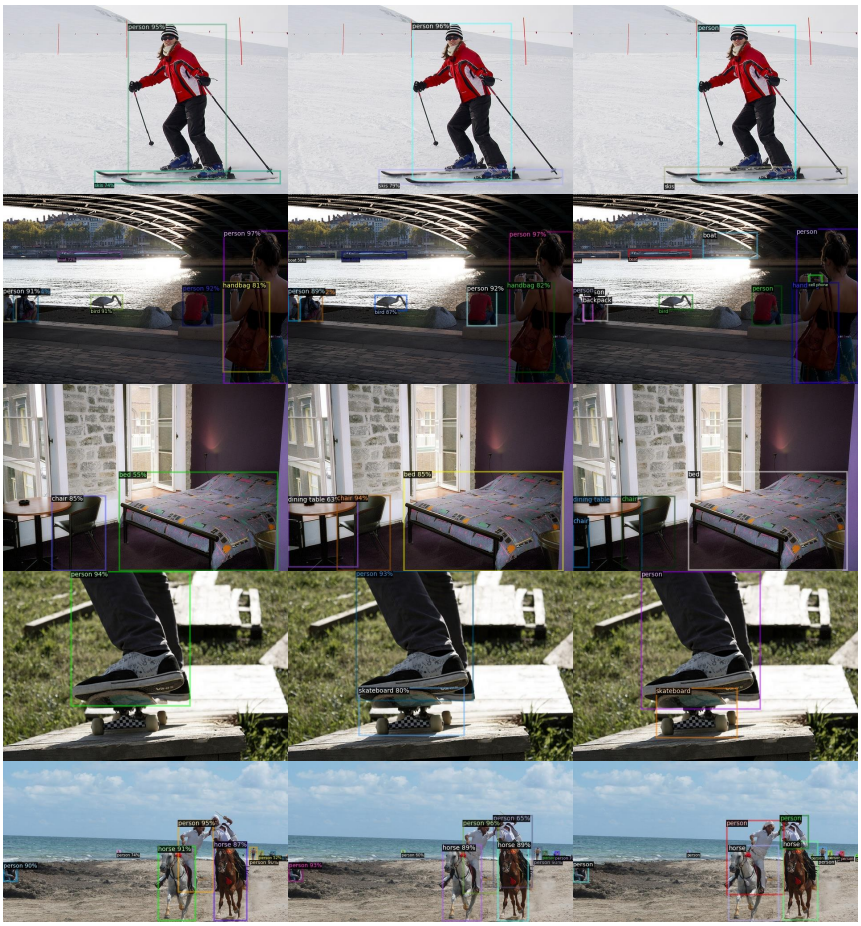
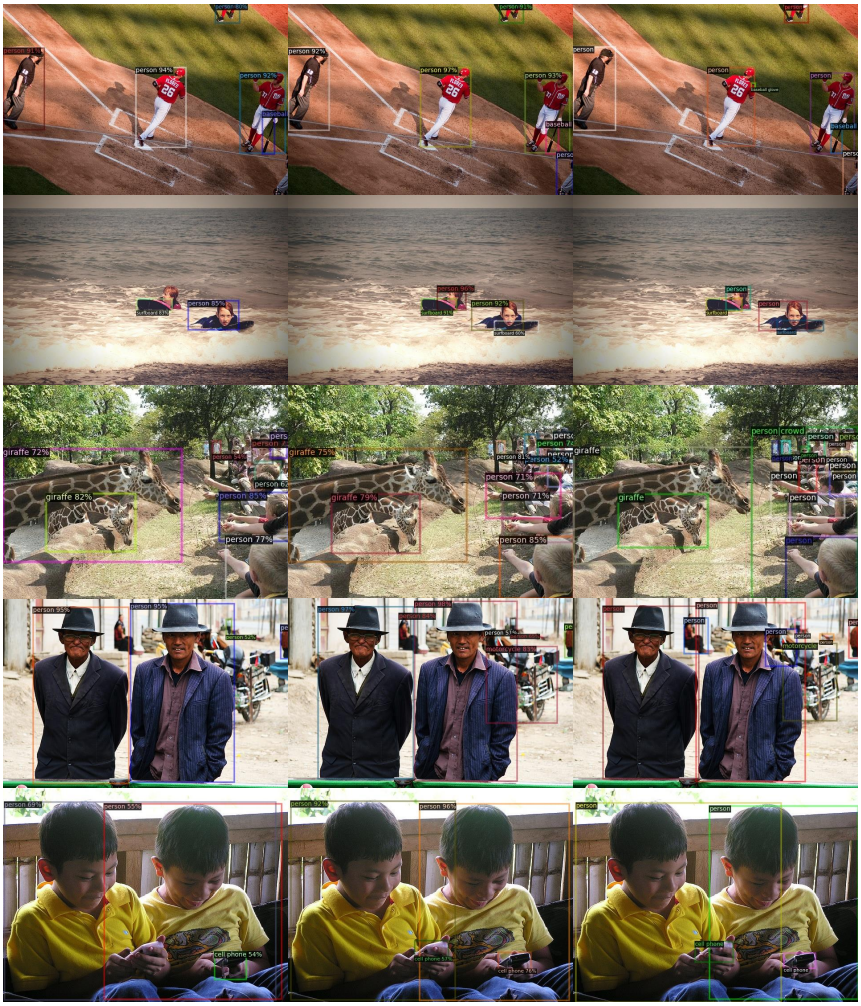


Fig. 2. Visualizations of detection results on COCO from sparse R-CNN, ours and ground truths. Our work is more better in complex scenarios.

Sparse R-CNN **ours** **GT**

Fig. 3. Visualizations of detection results on COCO from sparse R-CNN, ours and ground truths.



Sparse R-CNN **ours** **GT**

Fig. 4. Visualizations of detection results on COCO from sparse R-CNN, ours and ground truths.

Fig. 5. Visualizations of detection results on CrowdHuman from sparse R-CNN, ours and ground truths.