method	backbone	query type	epochs	AP	AP^S	AP^M	AP^L	#params.	FLOPs	fps
Mask2Former [7]	R50	100 queries	12	38.7	18.2	41.5	59.8	44M	226G	9.7
Fixed matching scheme	R50	100 queries	12	36.0	15.4	38.6	57.2	44M	226G	9.7
Auxiliary loss	R50	100 queries	12	38.2	17.7	41.1	59.9	44M	226G	9.7

Table 10. The results of the naive solutions on COCO Instance segmentation. All methods in the table are trained for 12 epochs. Both solutions do not work.

A. Disscussion on noise types

A.1. Why MP-Former works even without noise?

According to Table 8, MP training improves AP by 0.9 even without any noise, while in DN-DETR, the auxiliary denoising loss only works when the noise scale is > 0. This difference is due to two reasons:

1. DN-DETR adopt layer-by-layer refinement. It feeds GT boxes as the initial anchors. Then, it predicts offset $(\Delta x, \Delta y, \Delta w, \Delta h)$ in each layer and add it into the last layer's anchor. Without noises, the model can easily predict the GT by letting $(\Delta x, \Delta y, \Delta w, \Delta h) = (0, 0, 0, 0)$. Therefore, the model learns nothing when giving GT boxes. However, MP-Former feeds GT masks as attention masks which are not used to construct predictions. Therefore, even when GT masks are given, it is challenging to predict GT masks.

2. Even though we do not give it noises, MP-Former has noises in itself. GT masks are interpolated into the resolution of the corresponding feature map before being fed into a decoder layer. The interpolation induces noises.

A.2. Why shifting and scaling noises are worse than point noises?

In this section, we explain why shifting and scaling noise do not work for MP-Former while they are useful for DN-DETR. In short, the reason is that DN-DETR uses deformable attention to probe image features within a box while MP-Former uses standard attention to probe image features according to a mask. The different approaches to query features lead to their need for different approaches to control the noise magnitude.

Following, we take shifting noise as an example to illustrate the reason in detail. Before explaining the detailed reason, we have to figure out why we add noise. Actually, the core idea of MP-Former is to give strong guidance to help the model learn ground-truth (GT) masks easier. The strongest guidance should be the GT masks, but it is easy for the model to learn GT masks when given GT masks. Therefore we add noises to make the prediction harder. Note that the noises should be small, otherwise the noised GT mask provides no guidance. The magnitude of the noises can be measured by the IoU of the noised masks (boxes) and GT masks (boxes). Small noises correspond to large IoU. Fig. 4 shows examples of adding point and shifting noises on masks and adding shifting noise on boxes. Assuming we add point noises with 0.1 as the noise ratio, the IoU will be as follows.

$$\frac{1-0.1}{1} \leq IoU \leq \frac{1}{1+0.1} \qquad (4)$$
$$0.9 \leq IoU \leq 0.91$$

If we add a shifting noise of 0.1 to the mask, the IoU is indeterminate. In the example shown in Fig. 4 (b), the IoU is < 0.7. For shifting noises on masks, the IoU is dependent on not only the noise scale but also the shapes of the masks.

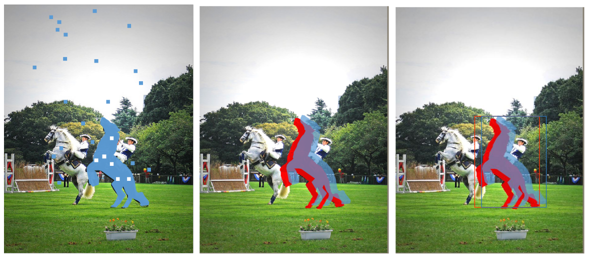

Figure 4. This figure is to help illustrate the reason why point noises work in MP-Former while shifting noises do not work. Red and blue regions are GT masks and noised masks, respectively. (a) Point noises on masks. (b) Shifting noises on masks (c) Shifting noises on boxes. Red and blue boxes are GT boxes and noised boxes, respectively.

However, when we add a shifting noise to a box, the IoU between a GT box and the noised box is determined. Assuming the noise scale is λ, we have $IoU = \frac{1-\lambda}{1+\lambda}$.

B. Implementation details

The number of MP groups: Each image contains a different number of objects. Adopting a fixed number of MP groups is unreasonable. For images with few objects, the GPU memory will be wasted, and for those with many objects, memory may not be enough. To maximize the utilization of queries in the MP part, we adopt a dynamic number of MP groups. We adopt a fixed number of queries and the number of MP groups changes according to the number of objects in the image. Let n_q, n_g, n_o denote the number of MP queries, MP groups, and objects. We have $n_g = \lfloor n_q / n_o \rfloor$.

Self-attention mask: There are self-attention and cross-attention in Transformer decoder. Self-attention is applied among decoder queries. We find that queries in MP part bring key information about the GT masks. When doing self-attention, queries in the matching part will simply copy MP queries and learn nothing. Therefore, we adopt a self-attention mask to stop information leakage from the MP part to the matching part. This information leakage also exists among different MP groups, so we also adopt attention masks to stop inter-group information.

Other details: Our models are trained on NVIDIA Tesla A100 GPUs with 40GB memory. All the models are trained with a total batch size 16, an initial learning rate 1.0×10^{-4}, and a multi-step learning rate scheduler to drop twice by 0.1 each. On ADE20k, our models with R50 and Swin-L backbone are trained on 2 and 4 GPUs, respectively. On Cityscapes, we train the model with 4 and 8 GPUs for our models with R50 and Swin-L backbone, respectively. On COCO2017, our models with R50 and R101 backbone are trained on 4 and 8 GPUs, respectively. All the hyper-parameters we do not mention are the same as Mask2Former.

C. Some naive solutions for inconsistent mask predictions

Before proposing our MP-Former , We have tried some naive approaches to resolve inconsistent mask predictions. We introduce two of them and show the experimental results.

1. **Fixed matching scheme:** A most direct approach to improve the $Util^i$ proposed in section 3 is to use the same matching scheme for all layers. Therefore, we propose to only do bipartite matching for the last decoder layer and let the previous layers all use the same matching scheme.

2. **Auxiliary loss:** A direct to improve mIoU-L^i is to add a loss to encourage large mIoU-L^i. We formulate the loss as follows.

$$L = -\sum_{i=1}^{9} \text{mIoU-L}^i \tag{5}$$

Note that mIoU-L^i is non-derivative. Therefore, we use mask loss of adjacent layers to approximate it as follows.

$$L = \sum_{i=1}^{9} L_{mask}(M_i, M_{i-1}) \tag{6}$$

Where $L_{mask}(M_i, M_{i-1})$ denotes the mask loss with M_i as prediction and M_{i-1} as label.

Table 10 shows the performance of above mentioned naive solutions. We find fixed matching scheme does not work. When we observe the matched queries during training we find that only a part of queries is matched and optimized most of the time. So, we guess it is necessary for different layers to have different matching in the matching part. We also find directly enlarging the mIoU with an extra loss function does not work. Because the loss will encourage the predicted mask to follow the mask in the previous layer. Since the predicted masks in the first few layers are of low quality, the subsequent masks following the previous ones will also be bad.

D. Differences with DN-DETR:

How to feed GT into decoder: We had some initial attempts to adopt DN-DETR's way to feed GT through decoder queries. Since DN-DETR feeds GT boxes as positional queries to the Transformer decoder, we also tried to map noised GT masks into fixed-length position queries. The queries should contain part of the information about GT masks so that the model knows where to focus on to recover the GT masks. The first way we tried is to use the average of the image features within GT mask as the decoder queries of denoising part adding noises to the GT masks where we take features or directly add noises to the averaged vectors. However, it did not work. We find it difficult for the model to recover GT masks from averaged queries because taking averages may lead to losing much useful information about the GT masks. Another way we have tried is to map flattened 16×16-resolution GT masks to position queries with an MLP. It still did not work. It seems difficult for the model to probe features within the GT masks as we want. Therefore, the main difference between our method and DN-DETR is that we feed GT masks as attention masks of the cross attention in Transformer decoder while DN-DETR feeds noised GT boxes as queries into the decoder. The root cause of the difference is the different attention mechanisms we are using. DN-DETR adopts deformable attention which samples positions around given reference points. While our method adopts standard attention which cannot fully use the position queries interpreted from GT masks. Fig. 5 shows the attention map when we feed the mask as the query to probe features comparing with the sampled points by DN-DETR. It shows that it cannot find the right position to focus on when adopting the above-mentioned way.

Multi-layer noises: In addition, we find adding mask guidance to multiple layers can further improve the performance as shown in Table 8. The more layers we add noised masks, the better the performance becomes. However, adding noised boxes to multiple layers does not work for DN-DETR. We visualize the output masks and boxes of the first decoder layer of our method and DN-DETR respectively in Fig 5(d)(e). we find that the output of DN-DETR is close to the GT boxes but some output masks of our model are far from GT masks. Therefore, giving mask guidance in multiple layer can further guide decoder layers to probe features. The root cause of this difference lies in the different ways of updating predictions.

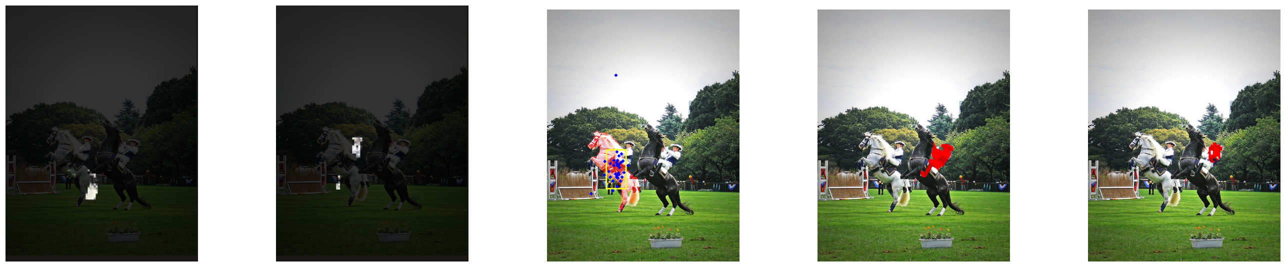

Figure 5. (a)(b) The attention map of the two ways to feed GT masks into decoder queries. (c) The blue points are the sampled locations in deformable attention and the yellow box is the predicted bounding box. Note that the corresponding GT object is the white horse. (d)The red region is a given GT mask in first decoder layer which covers the man riding the black horse.(e)The predicted mask of the first decoder layer which only covers the head of the man. The predicted mask will be used as the attention mask of the second layer and may mislead the second layer.

DN-DETR adopts layer-by-layer refinement. It predicts an offset $(\Delta x, \Delta y, \Delta w, \Delta h)$ in each layer and adds it into the last layer's prediction, which makes sure their box will not change much between layers. Therefore, when noised GT boxes are given in the first layer, the predictions of the following layers are usually close to GT boxes. Differently, Mask2Former adopts dot product to rebuild predictions in each layer. Therefore, it is more likely that their masks suffer great changes among layers.

Noise types: Finally, we find the shifting and scaling way of adding noises in DN-DETR does not suit our method well. As shown in Table 8, point noises suit our method best. The potential reason and better ways to add noise will be left for future work.