CVPDL HW1

B09901142 EE4 呂睿超

October 2024

Object Detector architecture

1. The model architecture/structure that I chose is DINO, the structure is as the following

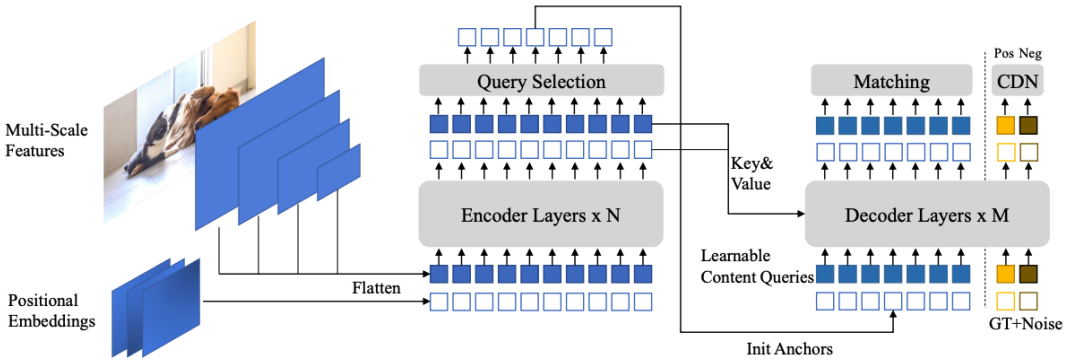

Fig. 2. The framework of our proposed DINO model. Our improvements are mainly in the Transformer encoder and decoder. The top-K encoder features in the last layer are selected to initialize the positional queries for the Transformer decoder, whereas the content queries are kept as learnable parameters. Our decoder also contains a Contrastive DeNoising (CDN) part with both positive and negative samples.

Figure 1: DINO structure with description

Implementations details

1. Model details
   (a) Basically, I adopted the exact implementation from the DINO repository and only do some detail parameter tuning following the instructions it provided

(b) I chose the DINO 4-scale(36 epoch setting) from the DINO model zoo with Swin-L backbone.

i. Specifically, I used the pre-trained weight of the model and the backbone from the link the repository provided.

ii. I trained these pre-trained weight on our custom dataset for 9 epochs

2. data format

(a) The given annotations are in YOLO format, so I changed it to COCO-format to fit the requirements of DINO model

(b) For this part, I utilize a reference on GitHub to ensure its correctness
(Note: First, I tried myself by using the transformation formula of (x_center, y_center, w, h) to (x_min, y_min, x_max, y_max) but there might probably be minor mistakes because I failed on training and visualized its error annotation)

3. Inference

(a) First, because of GPU limit, when training and inferencing, I resized the image to smaller sizes but with fix ratio.

(b) Observing the result that the model would have many redundant detected box but with very low confidence score, I adopted the method to set a threshold to ensure that my result is with a relatively high threshold. Specifically, I set it to 0.25

4. Other details

(a) As for loss function and data augmentation, I didn't modify the original setting from the repository.

(b) learning rate = 1e-4

(c) Because I did my training on Colab, I need to downgrade cuda library and fix lots of environment issues. For implementation detail, you can refer to the code.

(d) Environment: Colab T4 GPU

Performance for validation set

1. I failed to record the whole training process, but still I recorded some of the results of the process as the following images.

(a) Epoch 3

```
Averaged stats: class_error: 14.29  loss: 3.9736 (3.9326)  loss_bbox_dn: 0.0000 (0.0000
Accumulating evaluation results...
DONE (t=9.10s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.450
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.665
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.487
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.021
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.166
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.490
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.370
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.647
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.688
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.049
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.321
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.737
Training time 8:11:44
```

Figure 2: Epoch 3 Result

(b) Epoch 4

```
Averaged stats: class_error: 14.29  loss: 4.0505 (3.9851)  loss_bbox_dn: 0.0000 (0.0000)
Accumulating evaluation results...
DONE (t=10.11s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.470
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.677
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.519
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.024
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.186
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.511
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.382
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.658
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.687
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.070
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.320
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.736
```

Figure 3: Epoch 4 Result

(c) Epoch 5

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.473
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.699
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.518
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.022
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.173
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.515
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.376
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.659
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.697
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.060
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.325
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.746
```

Figure 4: Epoch 5 Result

(d) Epoch 6

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.490
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.715
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.535
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.022
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.175
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.532
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.382
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.655
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.690
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.044
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.327
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.738
Training time 8:33:59
```

Figure 5: Epoch 6 Result

(e) Epoch 9

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.497
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.726
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.545
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.028
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.176
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.540
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.384
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.661
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.700
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.062
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.326
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.749
Training time 9:37:09
```

Figure 6: Epoch 9 Result

2. Specifically, I arranged the results(mAP, AP50, AP75) as a table.

3. The mAP in the table is directly from the result that the training code yields. But after the threshold method I adopted and mentioned in implementation detail, the final inference validation set mAP is 0.689

Epoch	mAP (0.50:0.95)	AP50	AP75
3	0.450	0.665	0.487
4	0.470	0.677	0.519
5	0.473	0.699	0.518
6	0.490	0.715	0.535
9	0.497	0.726	0.545

Table 1: Model performance across epochs

Visualization and discussion

1. The visualization of a image in the validation dataset

Figure 7: Visualization of the object detection result

2. Long tail effect
   (a) To be honest, I can only give some observations rather than formal discussion and experiments.
   (b) The classes of the whole dataset are mostly centralized on person, hands, face, ears, ... because that there are people in most images but each tool isn't in every image.
   (c) Therefore, when inferencing, I observed that it is more challenging for the model to predict the classes of some tools such as face-guard and earmuffs(Moreover, they are mostly at similar positions)