Supplementary– PnP-DETR: Towards Efficient Visual Analysis with Transformers

Tao Wang[1,3*] Li Yuan[4*] Yunpeng Chen[2] Jiashi Feng[4] Shuicheng Yan[4]
[1] Institute of Data Science, National University of Singapore [2] Yitu Technology
[3] Integrative Science and Engineering Programme, NUS Graduate School, National University of Singapore
[4] Department of Electrical and Computer Engineering, National University of Singapore
twangnh@gmail.com ylustcnus@gmail.com yunpeng.chen@yitu-inc.com
jshfeng@gmail.com shuicheng.yan@gmail.com

Computation Saving

Here we show the concrete computation saving by the abstraction scheme, assume the length of the full feature set is $L = HW$ and the fraction of abstracted feature length is $r = (N + M)/L$. As shown in first row of Tab. 1, for encoder, since the complexity of self-attention layers is $\mathcal{O}(L^2)$ and the complexity of other layers (projection layers, feed-forward layers, normalization, *e.t.c*) is $\mathcal{O}(L)$,we assume their actual computation cost is aL^2 and bL correspondingly. For the decoder, since the complexity of cross-attention is $\mathcal{O}(L^2)$, and the complexity of other parts is not related to the sequence length L, we assume their costs are cL and a constant O respectively.

Then with the abstracted feature set $\mathbb{F}^*$ as input, the computation cost of encoder self-attention is quadratically reduced to ar^2L^2, and the cost of other layers is reduced linearly to brL. For the decoder, the cross attention cost is reduced to crL, and the cost of other layers remains as O. The total computation of encoder compared to the original is

$$\frac{ar^2L^2 + brL}{aL^2 + bL} = \frac{ar^2L + br}{aL + b} \in (r^2, r) \tag{1}$$

With a larger sequence length L the rate is more close to r^2 and more computation is saved.

The total computation of decoder compared to original is

$$\frac{crL + O}{cL + O} = \in (r, 1) \tag{2}$$

With a larger sequence length L the rate is more close to r and more computation is saved.

More Implementations

Here we describe the implementation details about padding masks and position embedding. For the fine feature set, we use the same sampling order of poll sampler to gather the corresponding position embeddings and padding masks. For the coarse feature set, we set the masks to $False$ to indicate that they are not paddings and employ pseudo position embedding by linearly combining position embeddings of the remaining feature set with the aggregation weight.

	encoder		decoder	
input set	self-attn	o. layers	cross-attn	o. layers
$\mathbb{F}$	aL^2	bL	cL	O
$\mathbb{F}^*$	ar^2L^2	brL	crL	O

Table 1. Computation saving by the abstract feature set $\mathbb{F}^*$, compared to the full set $\mathbb{F}$. self-attn and cross-attn indicate the self-attention and cross-attention layers. o. layers denotes other layers except for the self-attention and cross-attention.

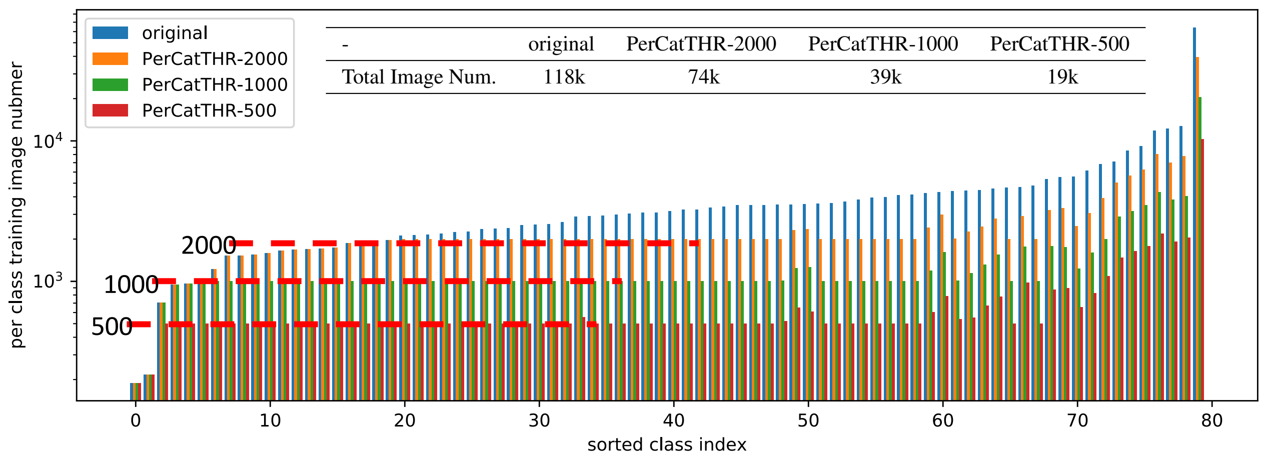
Class-Incremental Sampling on COCO Dataset

In this section, we present the detailed about how we sample the COCO dataset to obtain a smaller version for faster experimental validation. The COCO dataset has a skewed distribution of training image number over object categories, *i.e.*, some categories have significantly smaller number of training images. Direct random sampling on all training images may cause too much loss of images on those scarce categories and the overall distribution may be even more biased. The mAP result on the biased dataset may be unstable and cannot well evaluate the model performance. To curcumvent the difficulty and obtain more effective sampled dataset, we design a new strategy. We rank the object categories according to their training image number, then perform an incremental sampling starting from the most scarce category to the most abundant category. The the sampling algorithm is given in Algorithm 1. Concretely, for each category, if the number of training images is more than

*Work done during an internship at Yitu Tech.

Algorithm 1: Class-Incremental Sampling Algorithm.

```
Initialization:
Cat2ImgID: category to image id list mapping of original dataset (Dict);
PerCatTHR: per category sample image number threshold (Int);
Cat2ImgIdSampled: initialized to be same as Cat2ImgID (Dict);
SampledImgId: empty list (List);
SortedCatId: sorted category id on number of images, ascending (List);
$RandomSample$(Input,N): randomly sample input list to obtain subset with length N
for Id in SortedCatId do
    if Cat2ImgID[Id] $>$ PerCatTHR then
        InSampled = [ImgId for ImgId in Cat2ImgID[Id] if ImgId in SampledImgId];
        NotInSampled = [ImgId for ImgId in Cat2ImgID[Id] if ImgId not in SampledImgId];
        if len(InSampled) $<$ PerCatTHR then
            Cat2ImgIdSampled[Id]=InSampled+$RandomSample$(NotInSampled, PerCatTHR-len(InSampled))
        else
            Cat2ImgIdSampled[Id]=InSampled
        end
    end
    SampledImgId+=Cat2ImgIdSampled[Id]
end
```



-	original	PerCatTHR-2000	PerCatTHR-1000	PerCatTHR-500
Total Image Num.	118k	74k	39k	19k

Figure 1. Training image number distribution of the sampled COCO dataset obtained by the proposed class incremental sampling.

a sampling threshold number and the number of already sampled images for this category is less than the threshold number, then a sampling will be performed to obtain additional training images for reaching the threshold number. As shown in Fig. 1 is the distributions of obtained sampled versions of the COCO dataset, with different setting of the sampling threshold. The sampled dataset will be smaller given a smaller threshold. We use a sampling threshold of 500 to obtain a sampled COCO and conduct all the ablation experiments on the dataset. With the designed incremental sampling, the distribution of training images over most object categories is roughly uniform, and thus can be used to more stablly evaluate model performance than a randomly sampled sub-dataset while saving enormous experiment time.

Additional Ablations

Pool Sample Number M and Poll Sample Ratio α To individually examine the effect of M and α, we conduct following experiments: **1)** varying M by fixing α. As shown in Tab. 2, compared to the model with only poll sample feature vectors (M-0), adding 30 pool feature vectors gets about 1 AP improvement, but when M is larger than a certain value, the improvement is diminished (*i.e.*, 60). This phenomenon indicates that a small number of summarized feature vectors for the background contextual information is enough. **2)** varying α by fixing M. As shown in Tab. 3, when the

-	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
baseline	29.1	48.0	29.5	10.9	30.9	44.2
M-0	27.3	46.7	27.4	9.4	29.0	42.9
M-30	28.3	47.9	28.7	10.4	29.9	43.4
M-60	28.7	48.4	29.3	10.5	30.6	44.4
M-120	28.8	48.4	29.2	10.8	30.4	44.4

Table 2. Effect of pool sample number (M), with ResNet-50 backbone. The poll sample ratio α is fixed at 0.33.

-	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
baseline	29.1	48.0	29.5	10.9	30.9	44.2
α-0.1	25.2	44.7	24.2	8.1	26.0	40.9
α-0.2	27.1	46.4	27.3	9.6	29.0	43.0
α-0.3	28.7	48.4	29.3	10.5	30.6	44.4
α-0.5	29.2	48.6	29.2	10.8	30.3	44.1
α-0.7	29.1	48.2	29.3	11.0	30.9	44.1

Table 3. Effect of poll sample ratio (α), with ResNet-50 backbone. The pool sample number M is set as 60. The result is obtained with fixed poll ratio training.

poll ratio α is small, increasing it significantly improves the performance (*e.g.*, 25.2 AP to 27.1 AP by increasing α from 0.1 to 0.2). This observation shows the importance of fine information for detecting the objects. When α is larger than about 0.5, the performance improvement is diminished, which is as expected since the feature vectors that rank lower mostly correspond to the background locations, and thus the gain from including fine information on those locations is small.

Different Architecture of Scoring Network As shown in Tab. 4 is the result of different network architecture of the scoring network of the poll sampler, increasing the layer number from 1 to 2 improves the AP by 0.8 (*i.e.*, 1-layer-fc and 2-layer-fc-256.). This is likely because the 2-layer network much more accurately predict the informativeness score. Further increasing the layer number gives diminished gain, *i.e.*, 28.8 vs. 28.7 AP for 3-layer-fc-256 and 2-layer-fc-256. We also tried decreasing the hidden neuron unit number from 256 to 32, which reduces the computation, but the performance decreased, *i.e.*, 28.2 for the 2-layer-fc-32 scoring network, which is 0.6 lower than the 2-layer-fc-256 network in AP. We choose the 2-layer-fc-256 network as the default architecture of the score network.

Pool Sampler on The Full Feature Set While the proposed pool sampler operates on the non-sampled feature vectors of the poll sampler, it is interesting to see if directly applying the pool sampler on the full feature set for generating the coarse feature set would be better. As shown in Tab. 5, such setting leads to about 0.5 AP drop compared to the proposed two-step setting. This may be caused by the redundant information that have been captured by the fine

ScoreNet	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
1-layer-fc	27.9	47.5	28.3	10.0	29.6	43.2
2-layer-fc-256	28.7	48.4	29.3	10.5	30.6	44.4
3-layer-fc-256	28.8	48.4	29.4	10.3	30.7	44.6
2-layer-fc-32	28.2	48.0	29.1	9.9	30.4	44.0

Table 4. The effect of different scoring network architecture. For example, 1-layer-fc denotes 1-layer fully connected network (MLP), 2-layer-fc-256 means 2-layer fully connected network with 256 hidden neuron unit.

-	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
pool-after-poll	28.7	48.4	29.3	10.5	30.6	44.4
pool-full-set	28.2	47.5	28.7	10.5	29.9	43.2

Table 5. Result of applying pool sampler on the full feature set, compared to the proposed pool-after-poll design.

-	poll (proposed)	random	uniform	direct-interp
w/o pool	27.3	22.9	25.9	26.1
w pool	28.7	23.9	26.2	-

Table 6. Different alternative methods of the proposed poll sampling. The sampling ratio is set to 0.33 for all methods. w/o pool means removing the pool sampler. The random sampling result is obtained by an average of 3 runs.

feature vectors from polled samples.

Comparing the Proposed Sampling Strategies to Some Alternative Methods We compare the proposed poll sampler to some baseline alternatives including 1) random sampling: for each image, randomly sample the same amount of locations as the poll sampler and fix the sampled locations for training and evaluation. 2) uniform grid sampling: uniformly sample the 2D locations with equal interval. We adopt a general sampling mapping of $\lfloor \frac{i}{\sqrt{r}} \rfloor \lfloor \frac{j}{\sqrt{r}} \rfloor, i = 0, 1, ..., \lfloor W * \sqrt{r} \rfloor, j = 0, 1, ..., \lfloor H * \sqrt{r} \rfloor$ (H,W are the height and width of the feature map and r is the sampling ratio). With some specific poll ratio, the sampling is equavalent to MaxPooling, *e.g.*, $r = 1/4$ is equavalent to MaxPooling with kernel size 1 and stride 2. 3) direct interpolation: use interpolation to directly resize the feature map to target size $(\lceil H/\sqrt{r} \rceil, \lceil W/\sqrt{r} \rceil)$. As shown in Tab.6, compared to proposed ranking based poll sampling, random sample leads to a large drop in AP, *i.e.*, 22.9 vs 27.3 for the without pool sampling setting and 23.9 vs 28.7 for the with pool sampling setting. Uniform grid sampling and direct interpolation also generate lower performance than poll sampling, *e.g.*, 25.9 and 26.1 compared to 27.3 under the without pool sample setting. The result shows the proposed poll sampler learns effective sampling policy and is better than those simple baselines.