Container: Context Aggregation Network

Anonymous Author(s)
Affiliation
Address
`email`

Appendix

A Experimental setups

A.1 ImageNet Classification

ImageNet-1k is an image classification dataset with 1000 object categories. We use the basic architecture explained in Section **??**. All models are trained with the same setting as DeiT. Depthwise convolution, MLP and CONTAINER-LIGHT are trained with 8 16G V100 GPU with each GPU processing 128 images. Transformer and CONTAINER are trained with 8 80G A100 GPU and each GPU processes 128 images. Color jitter, random erase and mixup are used as data-augmentation strategies. We use the adamW optimizer. Learning rates are calculated using the following equation:

$$lr = \frac{lr_{base} \times Batch \times N_{GPU}}{512} \tag{1}$$

where base learning rate is chosen to be $5 \times e^{-4}$. We use cosine learning schedule and warm up the model in the first 5 epochs and train for 300 epochs in total.

A.2 Detection with RetinaNet

RetinaNet is a one-stage dense object detector using a feature pyramid network and focal loss. It is trained for 12 epochs, starting with a learning rate of 0.0001 which decreases by 10 at epoch 8 and 11. We use adamW optimizer and set weight decay to 0.05. No gradient clip is applied. We warm up for the first 500 iterations. Models are trained with 8 V100 GPU and each GPU holds 2 images. We freeze the batch normalization parameter similar to DETR.

A.3 Detection and Segmentation with Mask-RCNN

Mask-RCNN is a multi-task framework for object detection and instance segmentation. Mask-RCNN models are trained with 8 GPUs and each GPU hold 2 images. Mask-RCNN models are optimized by AdamW with a learning rate of 0.0001 and weight deacy of 0.05. We warm up the first 500 iterations. BN parameters are frozen for all layers.

A.4 Detection with DETR

DETR is an encoder-decoder transformer for end-to-end object detection. To improve the convergence speed and performance of DETR, SMCA-DETR propose a spatial modulated co-attention mechanism which can increase the convergence speed of DETR. Deformable DETR achieve fast convergence through deformable encoder and decoder. We compare CONTAINER-LIGHT with ResNet 50 on DETR without dilation, SMCA without multi scale and Deformable DETR without multi scale. DETR and SMCA DETR are optimized with 8 GPUs and 2 images per GPU, where as Deformable DETR uses 8 GPUs and 4 images per GPU. All models are optimzied with AdamW optimizer and weight clipping. DETR, SMCA DETR and Deformable DETR all use the default parameter setting in the original code release.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Method	Backbone	mAP	AP_S	AP_M	AP_L
DETR [1]	ResNet50	32.3	10.7	33.8	53.0
DETR [1]	CONTAINER-LIGHT	38.9	16.5	42.2	60.3
SMCA w/o multi-scale [2]	ResNet50	41.0	21.9	44.3	59.1
SMCA w/o multi-scale [2]	CONTAINER-LIGHT	44.2	23.8	47.9	63.1
DDetr w/o multi-scale [3]	ResNet50	39.3	19.8	43.5	56.1
DDetr w/o multi-scale [3]	CONTAINER-LIGHT	43.0	23.3	46.3	61.2

Table A.1: Comparison with DETR model over training epochs, mAP, inference time and GFLOPs.

A.5 Self-supervised Learning DINO

DINO is a recently proposed self-supervised learning framework. We adopt the default training setup in DINO to test the performance of CONTAINER-LIGHT on self-supervised learning. We compare with ViT-S/16 model using DINO. Baseline model and CONTAINER-LIGHT are trained using 100 epoches with cosine schedule for learning rate and weight decay. Learning rate at the end of warmup is 0.0005 while weight decay at the end will be kept constant to 0.4. Batch size per GPU is set to 64. We report kNN accuracy as a metric to evaluate the performance of self-supervised model.

B 1 line code change for Container-PAM

Listing 1: With just 1 line of code change in the forward pass of the Attention module within ViT, one can implement CONTAINER-PAM and obtain a +0.5 improvement on ImageNet top-1 accuracy.

```
class Attention(nn.Module):
    def __init__(self, dim, num_heads=8, qkv_bias=False, qk_scale=None, attn_drop
        =0., proj_drop=0., seq_l=196):

        super().__init__()
        self.num_heads = num_heads
        head_dim = dim // num_heads
        self.scale = qk_scale or head_dim ** -0.5

        self.qkv = nn.Linear(dim, dim * 3, bias=qkv_bias)

        #Uncomment this line for Container-PAM
        #self.static_a =
        #nn.Parameter(torch.Tensor(1, num_heads, 1 + seq_l , 1 + seq_l))
        #trunc_normal_(self.static_a)

        self.attn_drop = nn.Dropout(attn_drop)
        self.proj = nn.Linear(dim, dim)
        self.proj_drop = nn.Dropout(proj_drop)

    def forward(self, x):
        B, N, C = x.shape
        qkv = self.qkv(x).reshape(B, N, 3, self.num_heads, C // self.num_heads).
            permute(2, 0, 3, 1, 4)
        q, k, v = qkv[0], qkv[1], qkv[2]

        attn = (q @ k.transpose(-2, -1)) * self.scale
        attn = attn.softmax(dim=-1)

        #Uncomment this line for Container-PAM
        #attn = attn + self.static_a

        attn = self.attn_drop(attn)

        x = (attn @ v).transpose(1, 2).reshape(B, N, C)
        x = self.proj(x)
        x = self.proj_drop(x)
        return x
```

The attention code is borrowed from the TIMM library [1]. The one-line code addition in the forward pass for CONTAINER-PAM is implemented (and commented) in red. This code also requires enabling an additional parameter (also shown in red).

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 2

[2] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. *arXiv*, 2021. 2

[3] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 2

---

[1] `https://github.com/rwightman/pytorch-image-models/tree/master/timm`