SpatialFormer: Towards Generalizable Vision Transformers with Explicit Spatial Understanding Supplementary Material

Han Xiao[1,2*], Wenzhao Zheng[1,3*],
Sicheng Zuo[1], Peng Gao[2], Jie Zhou[1], and Jiwen Lu[1†]

[1]Tsinghua University [2]Shanghai AI Laboratory [3]UC Berkeley
https://wzzheng.net/SpatialFormer
xiaohan@pjlab.org.cn; wenzhao.zheng@outlook.com;
zsc23@mails.tsinghua.edu.cn;
gaopeng@pjlab.org.cn; {jzhou,lujiwen}@tsinghua.edu.cn

1 Vision Transformer Encoder and Decoder

We present the formulation of the Transformer Encoder and Decoder adopted in conventional vision transformers. In Vision Transformer, the Transformer Encoder is commonly utilized to learn representations from image patches and capture the relationships among them. The encoder's input is a sequence of 2D image tokens, obtained by dividing the input image into fixed-size patches and projecting them into patch embeddings. The Transformer Encoder consists of a stack of self-attention layers, including two key sub-modules: the self-attention mechanism and the feed-forward neural network. The self-attention mechanism enables the encoder to capture long-range dependencies between different patches in the input sequence. It calculates the weighted sum of input token features based on their pairwise attention. Specifically, given the input token feature Z, the self-attention (SA) mechanism can be formulated as follows:

$$[\mathbf{Q}, \mathbf{K}, \mathbf{V}] = \mathbf{Z}\mathbf{W}_{qkv}, \quad \mathbf{SA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V}, \tag{1}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$, n is the number of tokens, d is the dimension of features and $\mathbf{W}_{qkv}$ is a learnable matrix. Each Transformer Encoder takes the image tokens as input and process them throhgh the multi-head self-attention(MSA) and feed-forward networks(FFN):

$$\hat{\mathbf{Z}}^l = \mathbf{MSA}(\mathbf{LN}(\mathbf{Z}^{l-1})) + \mathbf{Z}^{l-1}, \quad \mathbf{Z}^l = \mathbf{FFN}(\mathbf{LN}(\hat{\mathbf{Z}}^l)) + \hat{\mathbf{Z}}^l, \tag{2}$$

where $\mathbf{Z}^l$ denotes the output token features of the l-th layer and $\hat{\mathbf{Z}}$ is an intermediate variable. LayerNorm (LN) layers and residual connections are applied after the feed-forward network within each encoder layer to generate the output.

* Equal contributions.
† Corresponding author.

Table 1: Configuration details of our SpatialFormer. P_i, C_i, and H_i denote the patch size, feature dimension, and num of heads of attention of Stage i, respectively.

	Output Size	Layer Name	SpatialFormer-T	SpatialFormer-S	SpatialFormer-B
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	Patch Embedding	$P_1 = 4$ $C_1 = 64$ $H_1 = 2$	$P_1 = 4$ $C_1 = 64$ $H_1 = 2$	$P_1 = 4$ $C_1 = 64$ $H_1 = 2$
		Transformer Decoder	$[\text{BCA}] \times 2$	$[\text{BCA}] \times 3$	$[\text{BCA}] \times 3$
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	Patch Merging	$P_2 = 2$ $C_2 = 128$ $H_2 = 4$	$P_2 = 2$ $C_2 = 128$ $H_2 = 4$	$P_2 = 2$ $C_2 = 128$ $H_2 = 4$
		Transformer Decoder	$[\text{BCA}] \times 2$	$[\text{BCA}] \times 4$	$[\text{BCA}] \times 6$
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	Patch Merging	$P_3 = 2$ $C_3 = 256$ $H_3 = 8$	$P_3 = 2$ $C_3 = 320$ $H_3 = 10$	$P_3 = 2$ $C_3 = 320$ $H_3 = 10$
		Transformer Decoder	$[\text{SA}] \times 6$	$[\text{SA}] \times 6$	$[\text{SA}] \times 18$
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	Patch Merging	$P_4 = 2$ $C_4 = 320$ $H_4 = 10$	$P_4 = 2$ $C_4 = 448$ $H_3 = 14$	$P_4 = 2$ $C_4 = 512$ $H_3 = 16$
		Transformer Decoder	$[\text{SA}] \times 2$	$[\text{SA}] \times 3$	$[\text{SA}] \times 3$

By stacking multiple layers of the aforementioned computation process, the Transformer Encoder effectively learns representations from image patches by modeling global dependencies between them. The final output of the encoder is typically employed for image classification by incorporating additional layers such as pooling and fully connected layers on top of the encoder.

For downstream tasks such as image segmentation and object detection, the sequence of embeddings obtained from the image encoder is usually fed into a decoder to generate an output sequence. The decoder, similar to the encoder, consists of a stack of identical layers, each comprising a self-attention mechanism and a feed-forward neural network. However, the decoder incorporates a third component, the cross-attention mechanism, which attends to the encoder's output and integrates it into the decoding process. Specifically, given the decoder's queries Q_d and the encoder's key and value embeddings K_e and V_e, the cross-attention (CA) between the encoder and the decoder is computed as follows:

$$\mathbf{CA}(\mathbf{Q_d}, \mathbf{K_e}, \mathbf{V_e}) = \mathbf{softmax}(\frac{\mathbf{Q_d}{\mathbf{K_e}}^T}{\sqrt{d}})\mathbf{V_e}, \tag{3}$$

Conditioned on the learned representations from the encoder, the Vision Transformer decoder generates the output sequence based on the specific task, such as segmentation masks and bounding boxes for object detection.

2 Configuration Details of SpatialFormer

We provide the detailed architectures of our SpatialFormer in Table 1. Following previous works [3, 8, 12], our models are constructed using a pyramid structure,

ensuring that the tiny, small, and base models have comparable parameters and FLOPs to existing methods. Table 1 showcases the comprehensive architectural specifications of our SpatialFormer model, providing a clear overview of its components and their configurations. The initial two stages of our model employ the proposed Bilateral Cross-Attention (BCA) Block, while the last two stages consist of Self-Attention (SA) blocks. This strategic arrangement allows for effective information exchange and integration between the 3D scene tokens and 2D image representations, enhancing the overall understanding of the 3D scene.

3 Dataset and Implementation Details

ImageNet Classfication. ImageNet [11] is a widely used large-scale benchmark for image classification, which contains around 1.2 million images from 1,000 categories. To implement our method, we use the PyTorch [10] framework and the timm library [13]. We train the models from scratch for 300 epochs with an input size of 224×224. We use the default data augmentation and regularization strategy, which includes MixUp [15], CutMix [14], and RandAugment [5] (rand-m9-mstd0.5-inc1). We adopt the AdamW optimizer with a cosine decay learning rate schedule and 10 epochs of linear warm-up. The initial learning rate, weight decay, and batch size are set to 0.001, 0.05, and 1024, respectively. Additionally, we finetune the SpatialFormer backbones at a resolution of 384×384 for 30 epochs following the settings in [9]. When fine-tuning the SpatialFormer backbones at the resolution of 384×384, we set the learning rate, weight decay, and batch size to be 5e-6, 1e-8, and 512, respectively.

Object Detection. To evaluate SpatialFormer for object detection and instance segmentation, we adapt existing frameworks (e.g., Mask-RCNN [6] and Cascade Mask R-CNN [1]) on the COCO 2017 dataset [7] as our benchmark. This dataset consists of 118K training images and 5K validation images with annotations for 80 object categories. We utilized SpatialFormer-S and SpatialFormer-B pretrained on ImageNet as the backbones and fine-tuned them on the COCO training set. We report the bounding box and mask Average Precision (AP^b and AP^m) at different IoU thresholds (50% and 75%).

Furthermore, we extend our evaluation to assess the versatility of SpatialFormer in developing transformer-decoder based detection frameworks. We report the mean Average Precision (AP^b) and Average Precision at different IoU thresholds (AP^b_{50} and AP^b_{75}) for three object sizes (small (AP^b_S), medium (AP^b_M), and large (AP^b_L)) for the object detection task.

Semantic Segmentation. We employ the ADE20K [16] dataset to evaluate the performance of SpatialFormer on semantic segmentation. ADE20K contains 20K training images and 2K validation images from 150 semantic categories. We implement the segmentation framework based on MMSegmentation [4].

In addition to incorporating SpatialFormer into the established UpperNet framework, we have developed a transformer-decoder based segmentation framework utilizing the output spatial tokens as initial object queries. To achieve this, we replicate the default 8×8 spatial tokens to a total of 256 and subsequently

select the top 100 tokens as object queries. We stack 9 transformer decoder layers to process the object queries and train the entire model with a set prediction loss following Mask2Former [2]. We report both the mean IoU (mIoU) and Multi-Scale (MS) mIoU on the validation set to evaluate the performance.

References

1. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: CVPR. pp. 6154–6162 (2018)
2. Cheng, B., Misra, I., Schwing, A.G., Kirillov, A., Girdhar, R.: Masked-attention mask transformer for universal image segmentation. In: PCVPR. pp. 1290–1299 (2022)
3. Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., Shen, C.: Twins: Revisiting the design of spatial attention in vision transformers. In: NeurIPS (2021)
4. Contributors, M.: Mmsegmentation: Openmmlab semantic segmentation toolbox and benchmark (2020)
5. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: PCVPRW. pp. 702–703 (2020)
6. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. pp. 2961–2969 (2017)
7. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV. pp. 740–755 (2014)
8. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
9. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. arXiv preprint arXiv:2201.03545 (2022)
10. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS. pp. 8026–8037 (2019)
11. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
12. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pvt v2: Improved baselines with pyramid vision transformer. Computational Visual Media **8**(3), 415–424 (2022)
13. Wightman, R.: Pytorch image models. `https://github.com/rwightman/pytorch-image-models` (2019). `https://doi.org/10.5281/zenodo.4414861`
14. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: ICCV. pp. 6023–6032 (2019)
15. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2018)
16. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. IJCV **127**, 302–321 (2019)