JUNE 18-22, 2023
CVPR
VANCOUVER, CANADA

BiFormer: Vision Transformer with Bi-Level Routing Attention

Lei Zhu[1], Xinjiang Wang[2], Zhanghan Ke[1], Wayne Zhang[2] and Rynson Lau[2]
[1]City University of Hong Kong, [2]SenseTime Research

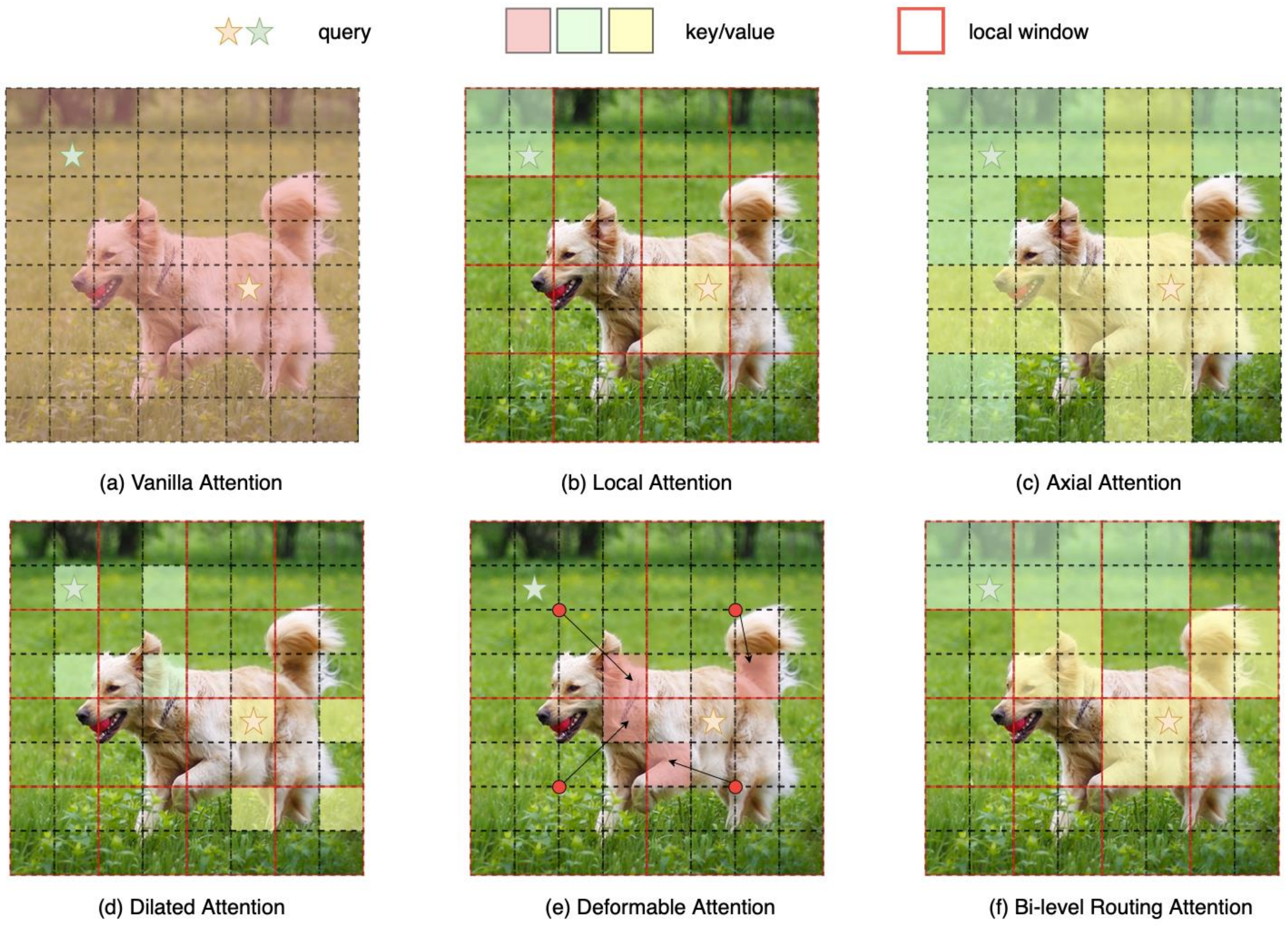
香港城市大學
City University of Hong Kong

sensetime

Outline

- Motivation
- Methodology
- Experiments
- Limitation & Future Work
- Conclusion

Motivation



- In the recent two years, **transformers** that originated from NLP have become the de facto standard architecture of state-of-the-art vision models.

- The core building block of transformers, **Scaled Dot-product Attention (SDA)**, is a double-edged sword:
  😊 Modelling long range dependency
  😊 Dynamic aggregation & in-context learning 😱 Quadratic computational complexity

- There are several works that introduced **sparse attention** to reduce the computation burden of SDA, however, they use
  - **Handcrafted** sparse pattern, such as (b) (c) (d)
  - Or **shared/query-agnostic** sparse sampling/down sampling, such as (e)

- Can we have **dynamic, query-aware** sparse SDA?

Motivation

DETR visualization :

- In fact, according to visualization of DETR (ECCV 2020) and ViT (ICLR 2021), the attention heatmap is
  - **Query-dependent**
  - **Non-local**

- We expect the sparse pattern better approximate the full attention

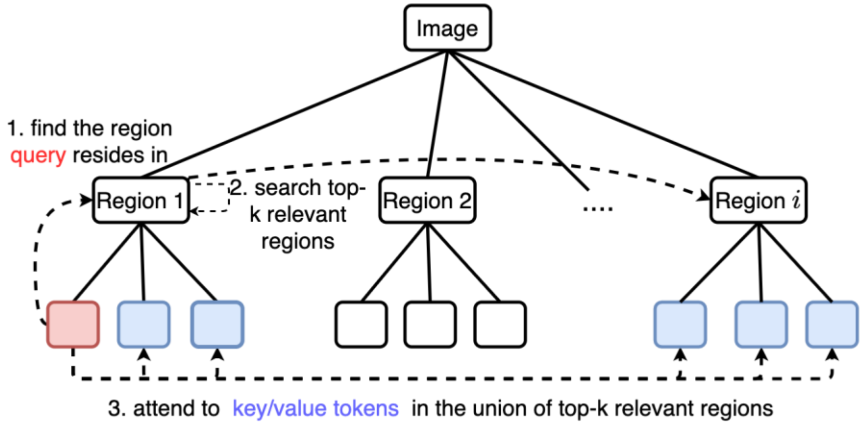
ViT attention visualization: https://epfml.github.io/attention-cnn/

Methodology

Algorithm 1 Pseudocode of BRA in a PyTorch-like style.

```
# input: features (H, W, C). Assume H==W.
# output: features (H, W, C).
# S: square root of number of regions.
# k: number of regions to attend.

# patchify input (H, W, C) -> (S^2, HW/S^2, C)
x = patchify(input, patch_size=H//S)

# linear projection of query, key, value
query, key, value = linear_qkv(x).chunk(3, dim=-1)

# regional query and key (S^2, C)
query_r, key_r = query.mean(dim=1), key.mean(dim=1)

# adjacency matrix for regional graph (S^2, S^2)
A_r = mm(query_r, key_r.transpose(-1, -2))

# compute index matrix of routed regions (S^2, K)
I_r = topk(A_r, k).index

# gather key-value pairs
key_g = gather(key, I_r) # (S^2, kHW/S^2, C)
value_g = gather(value, I_r) # (S^2, kHW/S^2, C)

# token-to-token attention
A = bmm(query, key_g.transpose(-2, -1))
A = softmax(A, dim=-1)
output = bmm(A, value_g) + dwconv(value)

# recover to (H, W, C) shape
output = unpatchify(output, patch_size=H//S)
```

bmm: batch matrix multiplication; mm: matrix multiplication. dwconv: depthwise convolution.



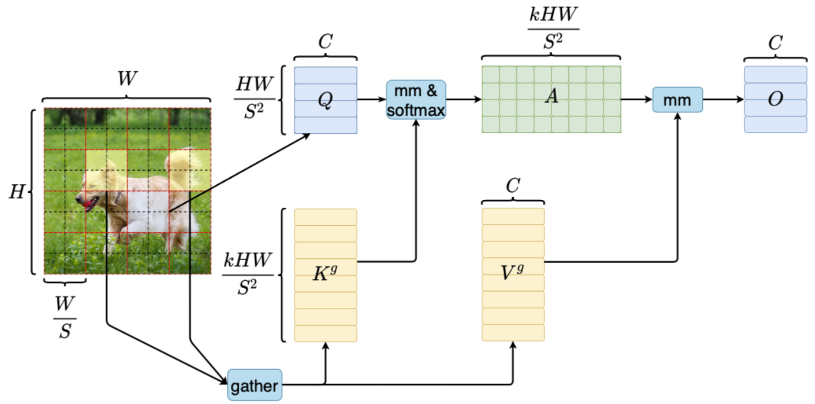


Figure 2. By gathering key-value pairs in top k related windows, we utilize the sparsity to skip computations in the most irrelevant regions, while only GPU-friendly dense matrix multiplications are involved.

- Coarse-to-fine scheme: Bi-level Routing Attention (BRA)
- Simple & GPU-friendly Implementation via key-value pair gathering
- Unlike masked attention, computation corresponding to zero-entries is skipped

Methodology

$$
\begin{aligned}
\mathrm{FLOPs} &= \mathrm{FLOPs}_{proj} + \mathrm{FLOPs}_{routing} + \mathrm{FLOPs}_{attn} \\
&= 3HWC^2 + 2(S^2)^2C + 2HWk\frac{HW}{S^2}C \\
&= 3HWC^2 + C(2S^4 + \frac{k(HW)^2}{S^2} + \frac{k(HW)^2}{S^2}) \\
&\geq 3HWC^2 + 3C(2S^4 \cdot \frac{k(HW)^2}{S^2} \cdot \frac{k(HW)^2}{S^2})^{\frac{1}{3}} \\
&= 3HWC^2 + 3Ck^{\frac{2}{3}}(2HW)^{\frac{4}{3}},
\end{aligned} \tag{8}
$$

The eqaulity holds if and only if

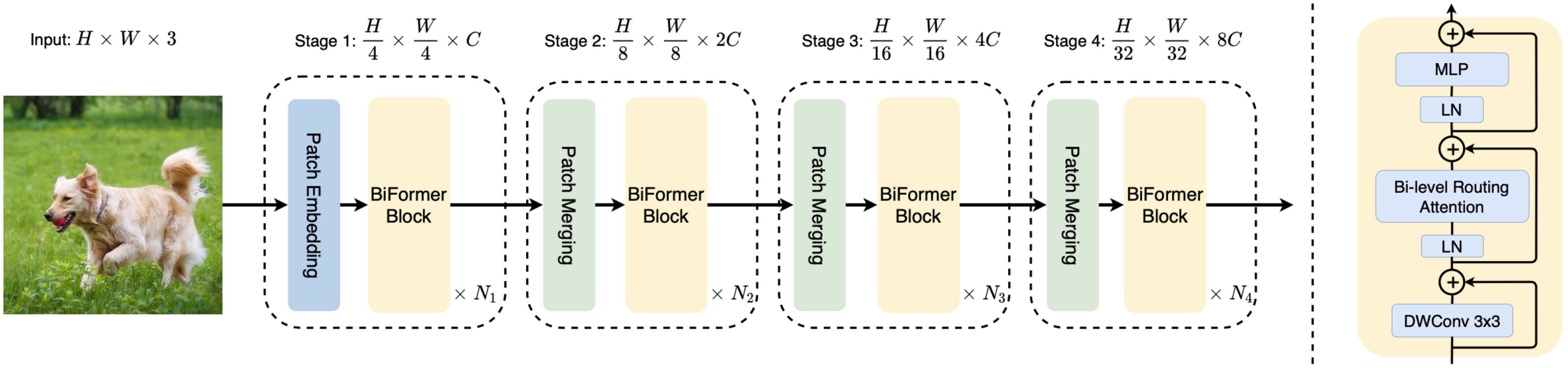
$$
S = (\frac{k}{2}(HW)^2)^{\frac{1}{6}}. \tag{9}
$$

Note: the complexity of (M, N, K) General Purpose Matrix Multiplication (i.e. M*K matrix multiply N*K matrix) is O(M*N*K)

Complexity analysis

- High-level intuition: lagrer S -> more regions & smaller region size -> FLOPs_routing↗ & FLOPs_attn↘, there is a trade-off
- Scaling S according to Eq.(9) in the paper achives best trade-offf, and achieves a complexity of $O((HW)^{\frac{4}{3}})$ as result.

Methodology



Models	#Channels.	#Blocks	Params	FLOPs
BiFormer-T	64	[2, 2, 8, 2]	13M	2.2G
BiFormer-S	64	[4, 4, 18, 4]	26M	4.5G
BiFomrer-B	96	[4, 4, 18, 4]	57M	9.8G

Table 1. Network width and depth of different model variants. The FLOPs are calculated with 224×224 input.

- We follow recent SOTAs to use a hierachical architecture design

Experiments

Model	FLOPs (G)	Params (M)	Top-1 Acc. (%)
ResNet-18 [18]	1.8	11.7	69.8
RegNetY-1.6G [32]	1.6	11.2	78.0
PVTv2-b1 [43]	2.1	13.1	78.7
Shunted-T [35]	2.1	11.5	79.8
QuadTree-B-b1 [36]	2.3	13.6	80.0
BiFormer-T	2.2	13.1	**81.4**
Swin-T [27]	4.5	29	81.3
CSWin-T [13]	4.5	23	82.7
DAT-T [45]	4.6	29	82.0
CrossFormer-S [44]	5.3	31	82.5
RegionViT-S [2]	5.3	31	82.6
QuadTree-B-b2 [36]	4.5	24	82.7
MaxViT-T [39]	5.6	31	83.6
ScalableViT-S [47]	4.2	32	83.1
Uniformer-S*	4.2	24	83.4
Wave-ViT-S* [48]	4.7	23	83.9
BiFormer-S	4.5	26	**83.8**
BiFormer-S*	4.5	26	**84.3**
Swin-B [27]	15.4	88	83.5
CSWin-B [13]	15.0	78	84.2
CrossFormer-L [44]	16.1	92	84.0
ScalableViT-B [47]	8.6	81	84.1
Uniformer-B* [24]	8.3	50	85.1
Wave-ViT-B* [48]	7.2	34	84.8
BiFormer-B	9.8	57	**84.3**
BiFormer-B*	9.8	58	**85.4**

Backbone	RetinaNet 1× schedule						Mask R-CNN 1× schedule					
	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	mAP^b	AP^b_{50}	AP^b_{75}	mAP^m	AP^m_{50}	AP^m_{75}
Swin-T [27]	41.5	62.1	44.2	25.1	44.9	55.5	42.2	64.6	46.2	39.1	61.6	42.0
DAT-T [45]	42.8	64.4	45.2	28.0	45.8	57.8	44.4	67.6	48.5	40.4	64.2	43.1
CSWin-T [13]	-	-	-	-	-	-	46.7	68.6	51.3	42.2	65.6	45.4
CrossFormer-S [44]	44.4	55.3	38.6	19.3	40.0	48.8	45.4	68.0	49.7	41.4	64.8	44.6
QuadTree-B2 [36]	46.2	67.2	49.5	29.0	50.1	61.8	-	-	-	-	-	-
WaveViT-S* [48]	45.8	67.0	49.4	29.2	50.0	60.8	46.6	68.7	51.2	42.4	65.5	45.8
BiFormer-S	45.9	66.9	49.4	**30.2**	49.6	61.7	**47.8**	**69.8**	**52.3**	**43.2**	**66.8**	**46.5**
Swin-S [27]	44.5	65.7	47.5	27.4	48.0	59.9	44.8	66.6	48.9	40.9	63.4	44.2
DAT-S [45]	45.7	67.7	48.5	30.5	49.3	61.3	47.1	69.9	51.5	42.5	66.7	45.4
CSWin-S [13]	-	-	-	-	-	-	47.9	70.1	52.6	43.2	67.1	46.2
CrossFormer-B [44]	46.2	67.8	49.5	30.1	49.9	61.8	47.2	69.9	51.8	42.7	66.6	46.2
QuadTree-B3 [36]	47.3	68.2	50.6	30.4	51.3	62.9	-	-	-	-	-	-
Wave-ViT-B* [48]	47.2	68.2	50.9	29.7	51.4	62.3	47.6	69.1	52.4	43.0	66.4	46.0
BiFormer-B	47.1	**68.5**	50.4	**31.3**	50.8	62.6	**48.6**	**70.5**	**53.8**	**43.7**	**67.6**	**47.1**

Backbone	S-FPN mIoU(%)	Upernet mIoU(%)	Upernet MS mIOU(%)
Swin-T [27]	41.5	44.5	45.8
DAT-T [45]	42.6	45.5	46.4
CSWin-T [13]	48.2	49.3	50.7
CrossFormer-S [44]	46.0	47.6	48.4
Shunted-S [35]	48.2	48.9	49.9
WaveViT-S* [48]	-	-	49.6
BiFormer-S	**48.9**	**49.8**	**50.8**
Swin-S [27]	-	47.6	49.5
DAT-S [45]	46.1	48.3	49.8
CSWin-S [13]	49.2	50.4	51.5
CrossFormer-B [44]	47.7	49.7	50.6
Uniformer-B [24]	48.0	50.0	50.8
WaveViT-B* [48]	-	-	51.5
BiFormer-B	**49.9**	**51.0**	**51.7**

- BiFormer have achieved decent performance compared to recent SOTAs on image classification, object detection and semantic segmentation

Experiments

Sparse Attention	IN1K Top1(%)	ADE20K mIoU(%)
Sliding window [33]	81.4	-
Shifted window [27]	81.3	41.5
Spatially Sep [6]	81.5	42.9
Sequential Axial [19]	81.5	39.8
Criss-Cross [21]	81.7	43.0
Cross-shaped window [13]	82.2	43.4
Deformable [45]	82.0	42.6
Bi-level Routing	**82.7**	**44.8**

Table 5. Ablation study on different attention mechanisms. All models follow the architecture design of the Swin-T model.

S	k	#tokens to attend	Acc	im/s (FP32)
7	1,4,16,49	64,64,64,49	**82.7**	522.3
7	1,2,8,32	64,32,32,32	82.4	563.2
7	2,8,32,49	128,128,128,49	82.6	419.9
8,4,2,1	2,2,2,1	98,98,98,49	82.3	**606.2**

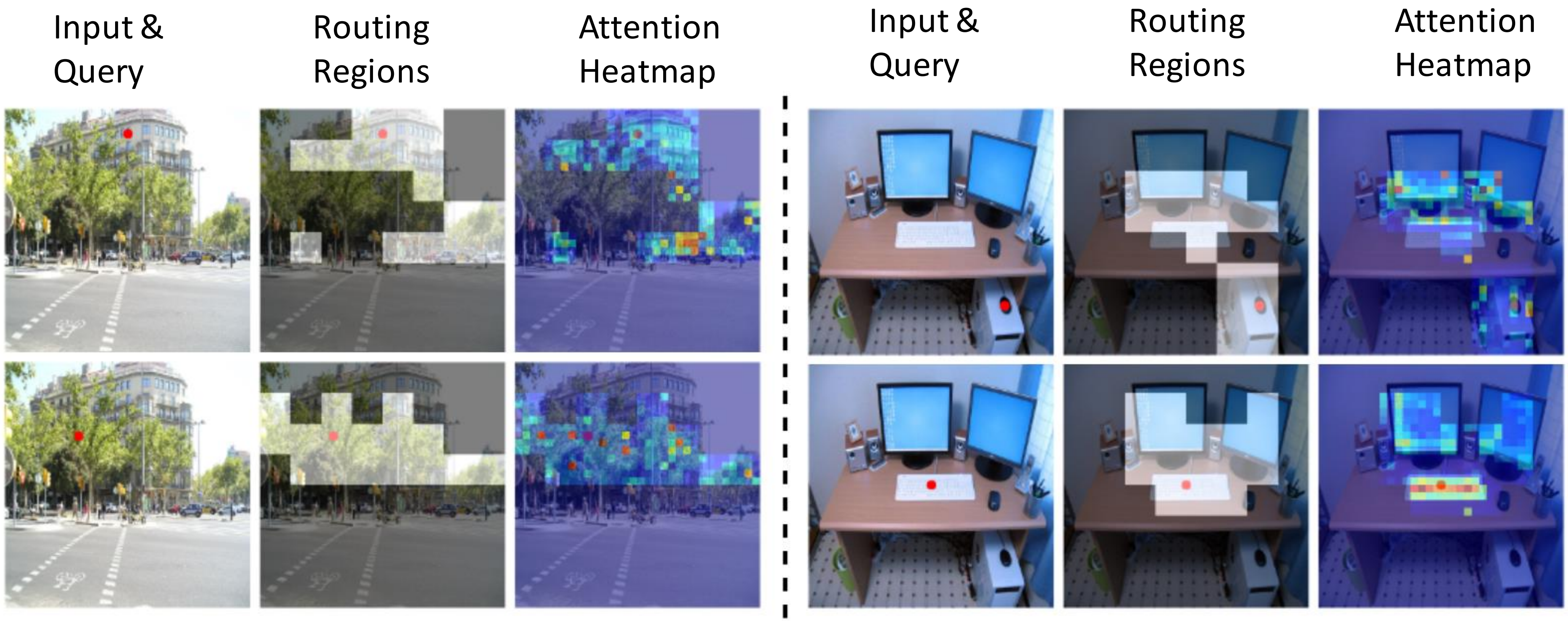
Table 7. Ablation study on top-k and partition factor S.

- Ablation studies
  - The proposed sparse attention (BRA) outperforms other sparse attention mechanisms
  - Increasing number of tokens to attend does not necessarily improve the performance, sometimes even hurts
  - Besides the attention mechanism, we also incorporate several architecture designs to achive SOTA performance

Architecture design	Params (M)	FLOPs (G)	IN1K Top1 (%)
Baseline (Swin-T layout)	29	4.6	82.7
+Overlapped patch emb.	31	4.9	82.8 (+0.1)
+Deeper layout	25	4.5	83.5 (+0.7)
+Convolution pos. enc.	26	4.5	83.8 (+0.3)
+Token Labling	29	4.9	84.3 (+0.5)

Table 6. Ablation path from Swin-T [27] layout architecture to BiFormer-S. Note that the modifications are applied sequentially.

Experiments



Limitation & Future Works

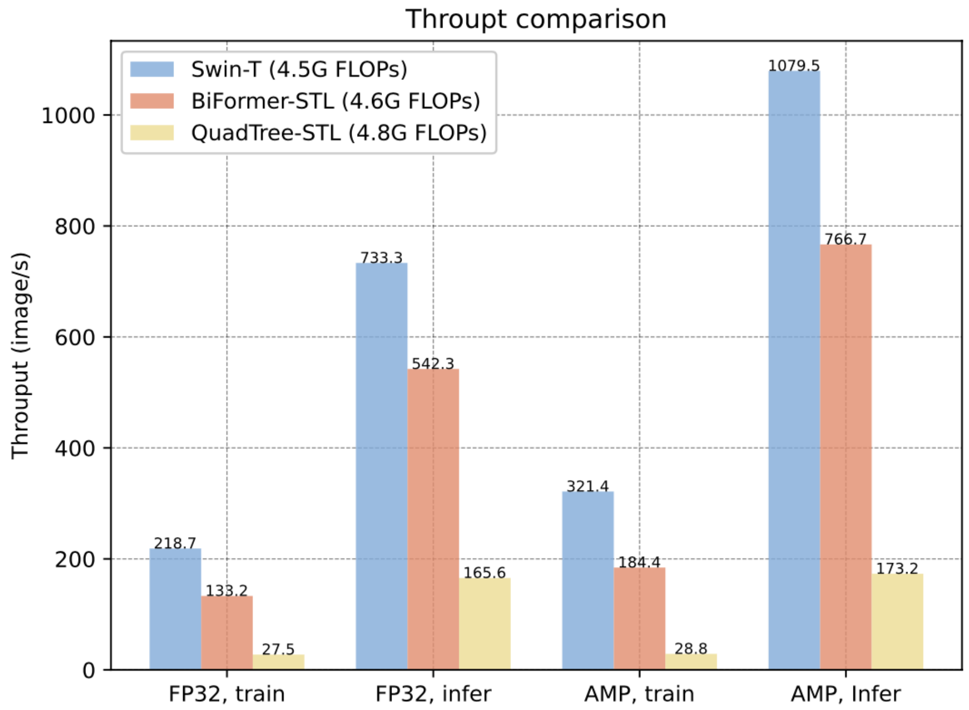


Figure 1. Throughput comparison on a 32GB Tesla V100 GPU. The suffix "STL" denotes **Swin-T Layout**, which means we use Swin-T [4] backbone with only attention module being replaced. We report results under both FP32 precision and automatic mixed precision (AMP) modes.

- Compared to sparse attention with handcrafted regular pattern (e.g. local window, axial stripes), BRA introduce an extra step to locate regions to attend (i.e. the routing step)
  - Small computation overhead (FLOPs)
  - More memory transactions (e.g. gather op)
  - More GPU kernel launches
- Can be improved via engineering efforts such as kernel fusion
- Besides, there are also some interesting research problems, e.g. when the sparsity can be **Ideally** utilized to accelerate the computation?
  - **Structured sparsity** instead of random sparsity
  - Arithmetic intensity matters

Conclusion

- We propose Bi-level Routing Attention (BRA), a novel **dynamic**, **query-aware** sparse variant of scaled dot-product attention. It can achieve a complexity of $O((HW)^{\frac{4}{3}})$ with proper region partition size.

- Using BRA as the core building block, we propose BiFormer, a family of vision transformers which achieve better FLOPs-Accuracy trade-off than existing sparse vision transformers.

- BRA does have a limitation: the throughput does not look as good as its FLOPs. Besides the engineering sides, there are also research opprtunities on how to sufficiently utilize the sparisty with **hardware-awareness**.

Paper

Code

More details can be found in our paper and code