Memory Efficient Transformer Adapter for Dense Predictions

Dong Zhang [1, 2] Rui Yan [3] Pingcheng Dong [1] Kwang-Ting Cheng [1, 2]

[1]The Hong Kong University of Science and Technology (HKUST); [2]AI Chip Center for Emerging Smart Systems (ACCESS); [3]Nanjing University (NJU)

Motivations & Solution

- The ViT adapter has emerged as a pivotal methodology for extracting vision-specific inductive biases from pre-trained ViT models[1], effectively mitigating the limitations inherent in the conventional pre-training followed by fine-tuning paradigm.
- While existing ViT adapters have demonstrated notable accuracy in vision tasks, their inference efficiency is substantially compromised by suboptimal memory access patterns[2], particularly due to operations such as standard normalization and frequent tensor reshaping.
- Our approach aims to mitigate memory access bottlenecks by strategically curtailing the dependence on layer normalization and significantly reducing frequent reshaping operations, ultimately enhancing the inference speed in downstream tasks.

Main Contributions

1. We introduced a memory-efficient Transformer adapter (META) block that leverages shared normalization operations across the self-attention and feed-forward network layers, which operate in a parallel manner[3].
2. Within this block, we employ the cross-shaped self-attention to minimize the dependency on frequent reshaping operations[4], thereby streamlining computational efficiency and reducing memory overhead.
3. To enhance local inductive biases for dense prediction tasks, we integrate a lightweight convolutional branch into the memory-efficient Transformer adapter block.
4. During interaction with the ViT backbone, a cascaded mechanism is introduced to compute multi-head features, thereby enhancing the diversity of the learned feature representations.
5. From a theoretical perspective, we also demonstrate that META exhibits superior generalization capability and enhanced adaptability when compared to existing ViT adapter methods.

Qualitative Superiorities

- As illustrated in Figure 1, qualitative performance evaluations across various models reveal significant advantages of the proposed META over existing ViT models and ViT adapters.
- META exhibits superior efficiency in terms of training parameter optimization, application gap reduction, memory access cost minimization, and inference time acceleration.
- These improvements highlight its effectiveness in bridging theoretical advancements with practical deployment requirements, making it a robust and scalable solution for vision transformer-based tasks.

	ViT	ViT Adapter	META(Ours)
Training Params.	☒ High	☑ Low	☑ Low
Application gaps	☒ High	☑ Low	☑ Low
Memory consumptions	☒ High	☒ High	☑ Low
Inference time costs	☒ High	☒ High	☑ Low

Figure 1. Qualitative performance comparisons of different models with respect to training parameters (Params.), application gaps, memory consumptions, and inference time costs.

META: Memory Efficient Transformer Adapter

As illustrated in Figure 2, the whole network mainly consists of two parts:

1. The upper part is a pre-trained plain ViT model, which consists of 4 backbone blocks.
2. The lower part is a trainable ViT adapter, which includes a spatial prior module and a set of **cascaded MEA injectors** and **cascaded MEA extractors** that act on each ViT backbone block.

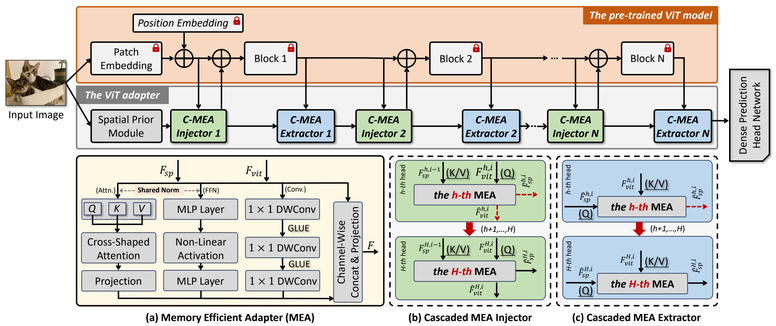

Figure 2. Overall architecture of META. Our primary contribution is the proposal of a MEA block in (a), which serves as the fundamental component for the injector in (b) and extractor in (c).

Memory Efficient Adapter (MEA) Block: Consider an arbitrary block in the proposed framework, where the input consists of two feature tensors: $\mathbf{F}_{sp}$ and $\mathbf{F}_{vit}$. The output feature representation, denoted as $\mathbf{F}$, is generated by concatenating the intermediate features extracted from the Attn, FFN, and Conv branches along the channel dimension. Subsequently, the concatenated feature is processed through a feature projection layer to produce the final output.

$$\mathbf{F} = \mathrm{Conv}_{3\times3}(\mathrm{Concat}(\underbrace{(\mathrm{A}(\mathbf{F}_{sp}, \mathbf{F}_{vit})}_{\textbf{Attn Branch}}; \underbrace{\mathrm{F}(\mathbf{F}_{sp}, \mathbf{F}_{vit})}_{\textbf{FFN Branch}}; \underbrace{\mathrm{C}(\mathbf{F}_{sp}, \mathbf{F}_{vit})}_{\textbf{Conv Branch}}; \mathbf{F}_{sp}; \mathbf{F}_{vit})), \quad (1)$$

The Attn Branch. To optimize memory efficiency and computational overhead, we employ cross-shaped self-attention in our framework[2].

$$\mathrm{A}^m_H(\mathbf{F}_{sp}, \mathbf{F}_{vit}) = \mathrm{SA}(\mathrm{LN}(\mathbf{F}^m_{sp} W^Q), \mathrm{LN}(\mathbf{F}^m_{vit} W^K), \mathrm{LN}(\mathbf{F}^m_{vit} W^V)), \quad (2)$$

The FFN Branch. The input of the FFN branch is $\mathbf{F}_{sp}$ and $\mathbf{F}_{vit}$.

$$\mathrm{F}(\mathbf{F}_{sp}, \mathbf{F}_{vit})_{\mathrm{Tem}} = \mathrm{LN}(\mathrm{Conv}_{3\times3}(\mathrm{Concat}(\mathbf{F}_{sp}; \mathbf{F}_{vit}))). \quad (3)$$

$$\mathrm{F}(\mathbf{F}_{sp}, \mathbf{F}_{vit}) = \mathrm{MLP}(\mathrm{NLA}(\mathrm{MLP}(\mathrm{F}(\mathbf{F}_{sp}, \mathbf{F}_{vit})_{\mathrm{Tem}}))), \quad (4)$$

The Conv Branch. The Conv branch also takes into $\mathbf{F}_{sp}$ and $\mathbf{F}_{vit}$ as the input.

$$\mathrm{C}(\mathbf{F}_{sp}, \mathbf{F}_{vit}) = \mathrm{DC}(\mathrm{GLU}(\mathrm{DC}(\mathrm{GLU}(\mathrm{DC}(\mathrm{Concat}(\mathbf{F}_{sp}; \mathbf{F}_{vit})))))). \quad (5)$$

MEA Injector & MEA Extractor

MEA Injector: $\mathbf{F}^i_{vit}$ is used as the query, and $\mathbf{F}^{i-1}_{sp}$ generated by the last extractor is used as the key and value. For each head, the output of the h-th head $\hat{\mathbf{F}}^{h,i}_{sp}$ and $\hat{\mathbf{F}}^{h,i}_{vit}$ is added into the input features of the next $(h+1)$-th head $\mathbf{F}^{h+1,i-1}_{sp}$ and $\mathbf{F}^{h+1,i}_{vit}$ to be used in the calculation of subsequent self-attention features, where $h = 1, 2, ..., H$.

MEA Extractor: $\hat{\mathbf{F}}^{h,i}_{sp}$ is used as the query, $\mathbf{F}^{h,i}_{vit}$ is used as the key and value. The output of the i-th cascaded MEA extractor is $\mathbf{F}^{h,i}_{sp}$.

Experiments & Results

Datasets: We conduct experiments on two datasets: MS-COCO for object detection (ODet) and instance segmentation (ISeg), and ADE20K for semantic segmentation (SSeg).

Baselines and settings: Mask R-CNN, Cascade Mask R-CNN, ATSS, and GFL are employed as the baseline models for ODet and ISeg. We select Semantic FPN and UperNet as baseline models for SSeg, where the Semantic FPN is trained for 80k iterations and the UperNet is trained for 160k iterations. All the baseline models are pre-trained on ImageNet-1k by default.

Methods	#P	FLOPs	MC	FPS	AP^b	AP^m	AP^m_{50}	AP^m_{75}
Cascade Mask R-CNN 3× +MS schedule								
Swin-T	86	745	NA	15.3	50.5	43.7	66.6	47.1
ViT-S	82	795	NA	16.5	47.9	42.8	62.1	44.8
ViT-Adapter-S	86	801	15.2	13.1	51.5	43.5	63.0	48.2
LoSA-S	84	799	13.0	12.0	53.5	43.8	64.1	48.1
META-S$_{(Ours)}$	83	797	8.1	14.7	54.3	44.8	66.8	49.5
Swin-B	145	982	NA	11.6	51.9	45.0	68.4	48.7
ViT-B	151	1,100	NA	13.0	50.1	42.7	63.9	44.1
ViT-Adapter-B	158	1,106	15.2	8.6	52.1	43.6	64.3	45.2
LoSA-B	155	1,104	13.0	9.0	52.6	43.5	63.8	44.7
META-B$_{(Ours)}$	153	1,101	8.1	11.9	53.8	44.1	65.2	45.6
ATSS 3× +MS schedule								
Swin-T	36	215	NA	17.1	47.2	41.2	54.8	45.5
ViT-S	32	263	NA	18.0	45.2	40.5	52.0	41.8
ViT-Adapter-S	36	272	15.2	14.2	49.6	42.5	55.1	46.5
LoSA-S	35	268	13.0	16.0	50.3	41.9	54.4	45.0
META-S$_{(Ours)}$	33	265	8.1	16.8	54.9	43.2	55.6	47.4
GFL 3× +MS schedule								
Swin-T	36	251	NA	17.5	47.6	40.8	54.2	46.0
ViT-S	32	275	NA	18.1	46.0	40.2	52.9	45.1
ViT-Adapter-S	36	288	15.2	15.3	50.0	43.1	54.5	47.1
LoSA-S	35	284	13.0	16.2	51.2	43.3	54.7	47.5
META-S$_{(Ours)}$	33	279	8.1	17.6	55.6	44.0	55.2	48.5

Table 1. Result comparisons with SOTA methods under Cascade Mask R-CNN, ATSS, and GFL on the *val* set of MS-COCO dataset for ODet and ISeg. MC: memory consumption.

ISeg and ODet results on the *val* set of MS-COCO									
ViT-B	Attn Branch.	FFN Branch.	Conv Branch.	Cascade	AP^m	AP^b	#P	FLOPs	MC	FPS
✓					41.3	45.8	113.6	719	NA	11.5
✓	✓				32.1	33.6	113.9	719	7.5	11.3
✓	✓	✓			43.4	46.5	114.4	719	7.5	11.3
✓		✓			30.8	31.2	114.0	719	0.4	11.4
✓	✓	✓	✓		44.1	48.1	115.3	719	7.5	10.2
✓	✓	✓		✓	43.6	49.3	114.4	720	8.1	11.0
✓	✓	✓	✓	✓	44.3	51.2	115.3	720	8.1	11.1

Table 2. Effectiveness of each component of META under Mask R-CNN. "NA" denotes not applicable.

References

1. Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. *Vision Transformer Adapter for Dense Predictions*. ICLR, 2022.
2. Otniel-Bogdan Mercea, Alexey Gritsenko, Cordelia Schmid, and Anurag Arnab. *Time-Memory-and Parameter-Efficient Visual Adaptation*. CVPR, 2024.
3. Bobby He and Thomas Hofmann. *Simplifying Transformer Blocks*. ICLR, 2024.
4. Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. *CSWin Transformer: A General Vision Transformer Backbone with Cross-Shaped Windows*. CVPR, 2022.