CARE Transformer: Mobile-Friendly Linear Visual Transformer via Decoupled Dual Interaction

Supplementary Material

6. Summary

We provide more details about our proposed method in the supplementary material due to the lack of space in the main body of the paper. We organize our supplementary material as follows:

- In Section 7, *i)* we summarize the configuration for the three types of our CARE Transformers, *i.e.,* CARE-S0, CARE-S1, CARE-S2; *ii)* we provide more details about the architecture of our CARE Transformers, including *CARE block*, *multiscale local inductive bias*, and *dual interaction module*.
- In Section 8, *i)* we provide more visualized comparisons between recent typical Transformer models and our CARE Transformers in terms of both accuracy and efficiency.
- In Section 9, we visualize the training loss of our CARE Transformers aiming to demonstrate that our asymmetrical decoupled learning framework can be easily optimized.
- In Section 10, we provide additional proofs for our asymmetrical decoupled learning strategy: *i)* ∋ **Proof 2** further demonstrates that our ∋ **Proposition 3** (in Section 3.3 of the main paper) still holds when learning local inductive bias in multiscale receptive field; *ii)* ∋ **Proof 3** extends ∋ **Proof 1** (in Section 3.3 of the main paper) to the version of using *multi-head* linear attention; *iii)* ∋ **Proof 4** demonstrates that when learning local inductive bias in multiscale receptive field and adopting *multi-head* linear attention at the same time, our ∋ **Proposition 3** remains valid.
- In Section 11, we discuss the limitations of our current study and outline our plans for future research.

7. CARE Transformers

We provide more details about the architecture of CARE Transformers in this section. We first elaborate the configuration about CARE-S0, CARE-S1, and CARE-S2 in Section 7.1. Then, we give the details of our CARE block, dual interaction module, and multiscale local inductive bias in Section 7.2

7.1. Configuration

We summarize the configuration for the three types of our CARE Transformers–CARE-S0, CARE-S1, and CARE-S2–in Table 6. The details are introduced in the following.

CARE-S0. The stem layer of CARE-S0 is implemented by using a 4×4 convolution with the stride set as 4. The numbers of CARE blocks are set as $\langle 2, 4, 8, 4\rangle$ for the four stages with the feature dimensions set as $\langle 24, 48, 96, 192\rangle$. We adopt a asymmetrical learning strategy and set the dimensions of local and global features as $\langle 16, 32, 64, 128\rangle$ and $\langle 8, 16, 32, 64\rangle$ respectively. The dimensions of the dynamic memory unit are set as $\langle 8, 16, 32, 64\rangle$ for the four stages. Following [42, 56], we do not apply linear attention to the first two stages of our models, as there exists more noisy in the shallow layers of neural networks but linear attention's high entropy property makes it difficult to suppress the influence of noise information. So, we utilize a 1×11 and a 11×1 depth-wise convolution to capture long-range information, which is also adopted in CARE-S1 and

Table 6. The configuration for our CARE-S0, CARE-S1, and CARE-S2. In the table, "DMU" denotes the dynamic memory unit. "Local Op" and "Global Op" denote the operations that are utilized to learn local inductive bias and long-range dependencies respectively.

Stages	Tokens	Layer Specification		CARE Transformers		
				CARE-S0	CARE-S1	CARE-S2
1	$\frac{h}{4} \times \frac{w}{4}$	Stem	Kernel	4×4 Convs, stride $= 4$		
			Dim	24	24	24
		DMU	Initializer	2×2 Convs, stride $= 2$		
			Dim	8	8	8
		CARE Block	Num	2	3	3
			Local Op	3×3 and 7×7 Depth-wise Convs		
			Global Op	1×11 and 11×1 Depth-wise Convs		
			Local/Global Dim	16/8	16/8	16/8
2	$\frac{h}{8} \times \frac{w}{8}$	Downsampling	Kernel	2×2 Convs, stride $= 2$		
			Dim	48	48	48
		DMU	Initializer	2×2 Convs, stride $= 2$		
			Dim	16	16	16
		CARE Block	Num	4	6	6
			Local Op	3×3 and 7×7 Depth-wise Convs		
			Global Op	1×11 and 11×1 Depth-wise Convs		
			Local/Global Dim	32/16	32/16	32/16
3	$\frac{h}{16} \times \frac{w}{16}$	Downsampling	Kernel	2×2 Convs, stride $= 2$		
			Dim	96	96	144
		DMU	Initializer	2×2 Convs, stride $= 2$		
			Dim	32	32	48
		CARE Block	Num	8	10	10
			Local Op	3×3, 7×7 Depth-wise Convs		
			Global Op	Linear Attention		
			Local/Global Dim	64/32	64/32	96/48
4	$\frac{h}{32} \times \frac{w}{32}$	Downsampling	Kernel	2×2 Convs, stride $= 2$		
			Dim	192	192	288
		DMU	Initializer	2×2 Convs, stride $= 2$		
			Dim	64	64	96
		CARE Block	Num	4	6	6
			Local Op	3×3 and 7×7 Depth-wise Convs		
			Global Op	Linear Attention		
			Local/Global Dim	128/64	128/64	192/96

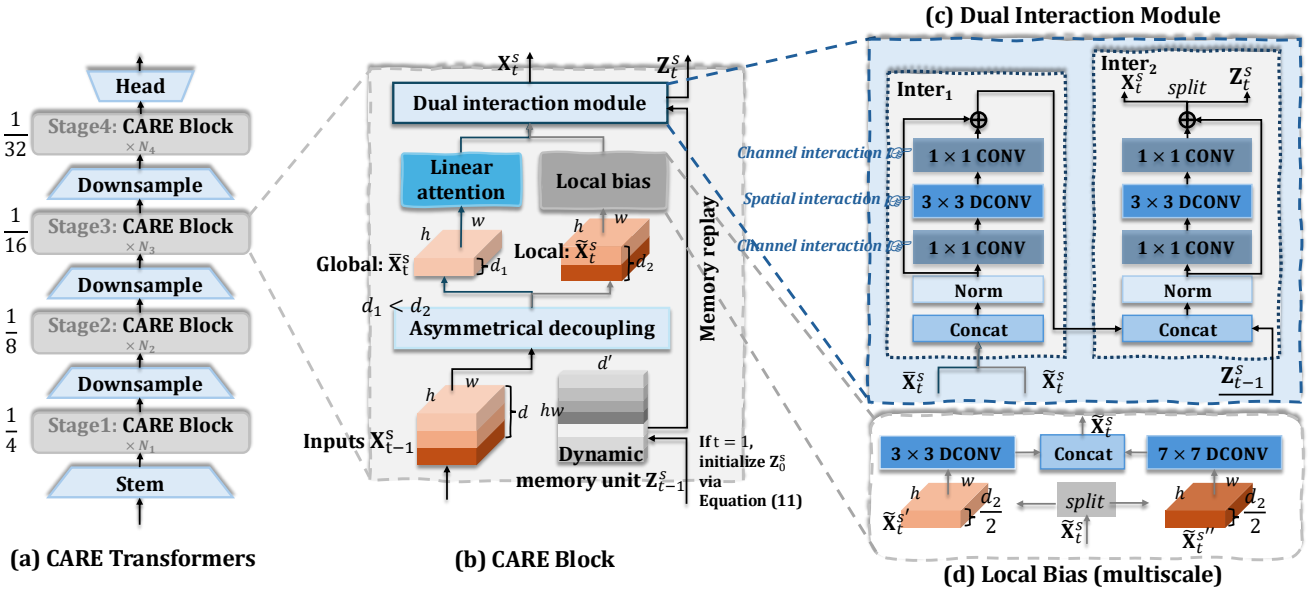


Figure 4. Architecture details of our CARE Transformers.

CARE-S2.

CARE-S1. We implement the stem layer of CARE-S1 by using a 4×4 convolution with the stride set as 4, which is the same as that in CARE-S0. The numbers of CARE blocks are set as $\langle 3, 6, 10, 6 \rangle$ for each stage and correspondingly the dimensions of features are set as $\langle 24, 48, 96, 192 \rangle$. Moreover, we set the dimensions of local and global features as $\langle 16, 32, 64, 128 \rangle$ and $\langle 8, 16, 32, 64 \rangle$ respectively, while the dimensions of the dynamic memory unit are set as $\langle 8, 16, 32, 64 \rangle$.

CARE-S2. We utilize a 4×4 convolution with the stride set as 4 to implement the stem layer of our CARE-S2. Moreover, the number of CARE blocks in each stage is set as $\langle 3, 6, 10, 6 \rangle$ with the feature dimensions set as $\langle 24, 48, 144, 288 \rangle$. We set the dimensions of local and global features in an asymmetrical manner as well, which are set as $\langle 16, 32, 96, 192 \rangle$ and $\langle 8, 16, 48, 96 \rangle$ for the four stages respectively. Additionally, for CARE-S2, we set the dimensions of the dynamic memory unit as $\langle 8, 16, 48, 96 \rangle$.

7.2. CARE block

The CARE block is the key component of our CARE Transformers. In this part, we give more details about our CARE block.

Multiscale Local Inductive Bias. In CARE Transformers, we learn local inductive bias in multiscale receptive field as described in Figure 4 (d). For the input of local features $\widetilde{\mathbf{X}}_t^s \in \mathbb{R}^{hw \times d_2}$, we divide them into two parts, and utilize a 3×3 and a 7×7 depth-wise convolution to process them respectively, which can be described by the following equations:

$$\widetilde{\mathbf{X}}_t^{s'}, \widetilde{\mathbf{X}}_t^{s''} = \text{Split}(\widetilde{\mathbf{X}}_t^s, dim{=}1), \tag{14}$$

$$\widetilde{\mathbf{X}}_t^{s'} = \text{DCONV}_{3\times3}(\widetilde{\mathbf{X}}_t^{s'}), \tag{15}$$

$$\widetilde{\mathbf{X}}_t^{s''} = \text{DCONV}_{7\times7}(\widetilde{\mathbf{X}}_t^{s''}). \tag{16}$$

Thereby, the model is endowed with the capability of being aware of multiscale local information. The ablation studies in Table 5 of the main paper indicate that learning local inductive bias in multiscale receptive field is better than using a 3×3 depth-wise convolution alone. In Section 10 of the supplementary material, we also provide ↄ **Proof 2** to demonstrate that when learning local inductive bias in multiscale receptive field, the asymmetrical decoupling strategy can still reduce the computational complexity of our models.

Dual Interaction Module. Furthermore, we also introduce the philosophy behind the design of our dual interaction module, where the key component is the interaction block (*i.e.,* **Inter**$_1$ and **Inter**$_2$) as exhibited in Figure 4 (c). In each interaction block, we consider both channel and spatial interaction aiming to facilitate information exchange between features.

We take the interaction block **Inter**$_1$ as an example to explain the interaction process. As illustrated in Figure 5, in **Inter**$_1$, we first conduct the channel interaction between local and global features by concatenating them together, and utilizing a 1×1 convolution to realize information propagation along channels and map the inputs to a high dimensional space. After fusing the information between local

and global features, we further conduct the spatial interaction by additionally applying a 3×3 depth-wise convolution to the learned high dimensional feature map, allowing the fused local and global features to interact with the information from neighboring pixels and thereby considering the local and global information in the spatial domain. At last, an extra 1×1 convolution is employed to perform the channel interaction again and map the features back to the original representation space. In this way, the information in the learned local and global features can be fully considered. The interaction process between the fused local and global features and the dynamical memory unit in $\textbf{Inter}_2$ is the same as that in the first interaction block.

Asymmetrical Setting of d_1 and d_2. In our CARE block, we adopt an asymmetrical setting $d_1 = \frac{1}{2}d_2$ (*i.e.*, $d_1 = \frac{1}{3}d$ and $d = d_1 + d_2$), where d_1 and d_2 represent the channel dimension of decoupled feature maps fed into linear attention and depth-wise convolution operations respectively. Our asymmetrical setting can obviously achieve better performance, *e.g.*, achieving 82.1% accuracy on ImageNet-1K and $2.0ms$ runtime latency per 224×224 image on iPhone13. However, setting $d_1 = d_2$ (*i.e.*, $d_1 = \frac{1}{2}d$) increases our model's latency to $2.2ms$ but does not improve the accuracy, indicating that our asymmetrical learning strategy can further boost the efficiency of models without largely sacrificing their accuracy. Moreover, setting $d_1 = d$ dcreases the accuracy to 77.3% and increases the latency to $2.5ms$, which demonstrates the importance of exploiting long-range dependencies in an asymmetrical manner.

8. Additional Visualized Comparisons

Our method is based on linear attention, and thus we provide more comparisons between our CARE Transformers and recent mobile-friendly visual Transformer models in this section, of which results are summarized in Figure 6, 7, 8, and 10. As NAS is not adopted to search for the optimal architecture of our models, we do not exhaustively compare CARE Transformers with methods using NAS algorithms. We leave NAS-based CARE Transformers for our future work.

8.1. ImageNet-1K

The comparisons between recent typical mobile-friendly Transformer models and our CARE Transformers on the image classification task of ImageNet-1K are summarized in Figure 6. We have the following observations for these results. At the same level of efficiency, our models clearly have higher accuracy. For example, at the cost of 0.7 GMACs, our CARE-S0 can achieve 78.4% top-1 accuracy, which is higher than that of FastViT-T8, SLAB-DeiT-T, EMO-1M (i.e., the smallest-size models of FastViT, SLAB,

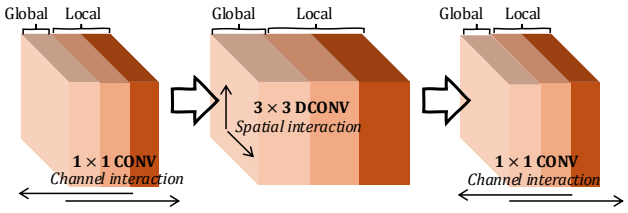


Figure 5. Schematic illustration for the channel and spatial interaction designed in the interaction block of our dual interaction module. In the figure, we take the interaction between local and global features (i.e., $\textbf{Inter}_1$) as an example to introduce the interaction process.

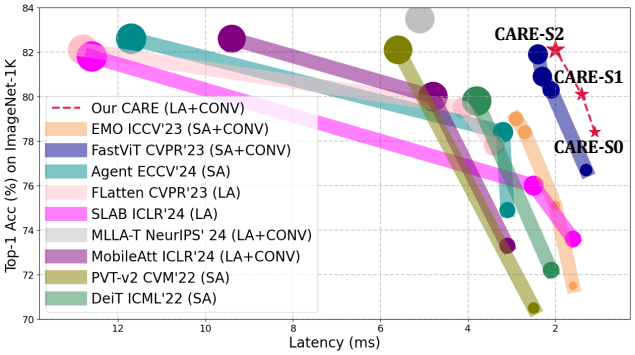


Figure 6. Visualized comparison of the balance between accuracy, latency, and GMACs for our CARE Transformers and recent typical mobile-friendly Transformer models on the image classification task of ImageNet-1K. In the figure, larger markers indicate that models need to consume more GMACs. "SA" and "LA" indicate that the methods are based on Self-Attention and Linear Attention respectively. The latency is tested on iPhone13 with the batch size set as 1 and the standard 224×224 input resolution.

and EMO in Figure 6) by 1.7%, 3.8%, and 6.9% respectively. Meanwhile, compared with the models having the same level of accuracy, our CARE Transformers have obviously higher efficiency. For example, our CARE-S2 is about $6\times$, $6\times$, $6\times$, $4\times$, and $2\times$ faster than FLatten-Swin-T, SLAB-Swin-T, Agent-Swin-T, MobileAtt-PVT-v2-B2, and PVT-v2-B2 (i.e., the largest-size models of FLatten, SLAB, Agent, MobileAtt, and PVT-v2 listed in Figure 6). Our CARE-S2 costs half the GMACs of MLLA-T and has $2.5\times$ faster speed on iPhone13, but its accuracy is only lower than that of MLLA-T by 1.4% (For more details about the comparisons between MLLA and our CARE Transformers, please refer to Section 8.4). These comparison results indicate that our CARE Transformers can achieve a better balance between accuracy and efficiency than the recent counterparts.

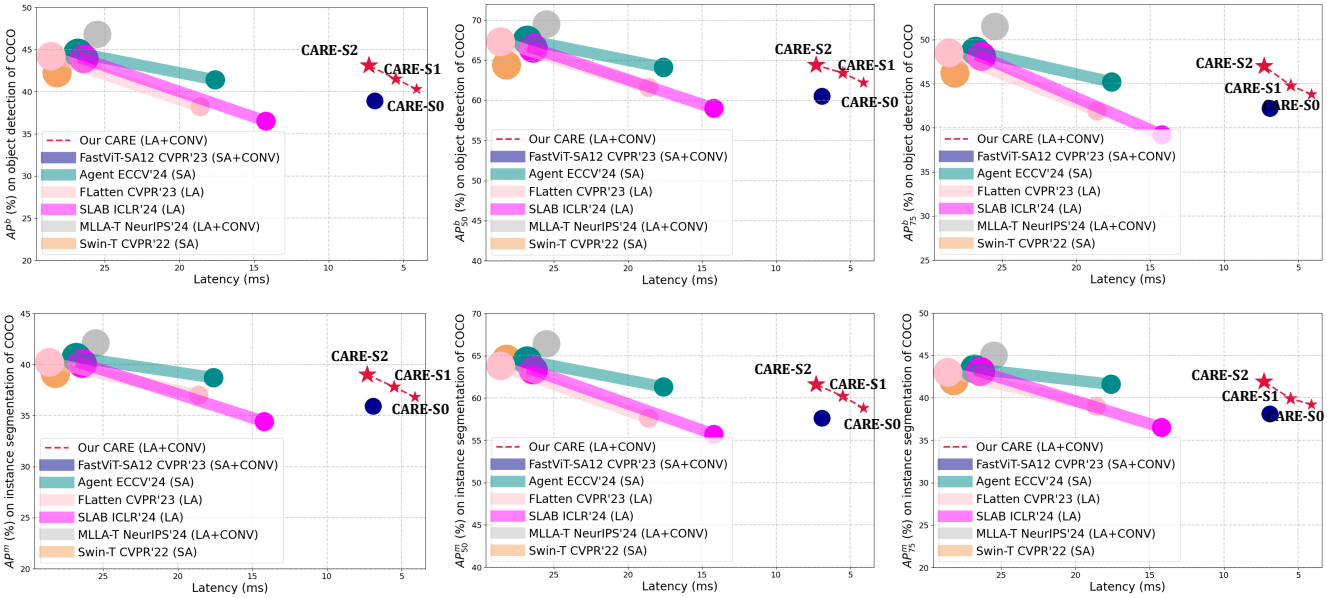


Figure 7. Visualized comparison of the balance between accuracy, latency, and GMACs for our CARE Transformers and recent typical efficient visual Transformer models on the object detection (*the first row*) and the instance segmentation task (*the second row*) of the COCO dataset. The larger the marker, the more GMACs the model consumes. "SA" and "LA" indicate that the methods are based on Self-Attention and Linear Attention respectively. The latency of models is measured on iPhone13 with the batch size set as 1 and the 512×512 input resolution. We build the detection head of our CARE Transformers by using Mask R-CNN [21].

8.2. COCO

The comparisons in Figure 7 demonstrate the superiority of our CARE Transformers in resource-constrained scenarios again. On the object detection task, our CARE-S0, CARE-S1, and CARE-S2 outperform FLatten-PVT-T and SLAB-PVT-T (*i.e.,* the smallest-size models of FLatten and SLAB listed in Figure 7) in both accuracy and efficiency. Although CARE-S2 has obviously less GMCAs, it can still achieve comparable accuracy to FLatten-Swin-T, Agent-Swin-T, SLAB-Swin-T, and Swin-T (FLatten-Swin-T, Agent-Swin-T, and SLAB-Swin-T are the largest-size models of FLatten, Agent and SLAB presented in Figure 7). Besides, CARE-S2 costs half of the GMACs of MLLA-T and has $3\times$ faster speed on iPhone13, yet its AP^b is lower than that of MLLA-T by only 3.7%, which is acceptable. Similar results can also be found on the instance segmentation task of the COCO dataset. For example, our CARE-S2 has obviously higher efficiency than Swin-T, Flatten-Swin-T, Agent-Swin-T, and SLAB-Swin-T (Flatten-Swin-T, Agent-Swin-T, and SLAB-Swin-T are the largest-size models of FLatten, Agent and SLAB listed in Figure 7) while still maintaining comparable accuracy to these counterparts.

8.3. ADE20K

The comparison results in Figure 8 also demonstrate the competitive performance of our CARE Transformers on the semantic segmentation task of ADE20K. Our CARE-S2 outperforms Swin-T, FLatten-PVT-T, and Agent-PVT-T (FLatten-PVT-T and Agent-PVT-T are the smallest-size models of FLatten and Agent listed in Figure 8) in both accuracy and efficiency. Despite that the mIoU of our CARE-

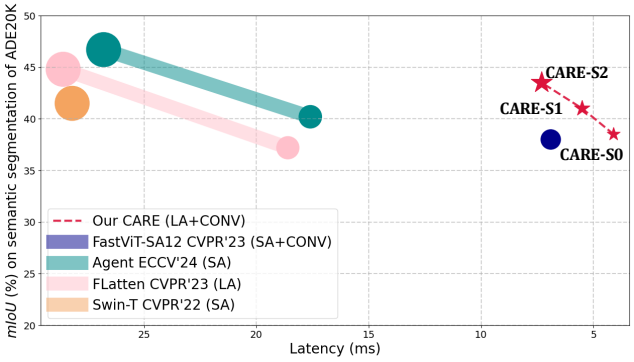


Figure 8. Visualized comparison of the balance between accuracy, latency, and GMACs for our CARE Transformers and recent typical Transformer models on the semantic segmentation task of ADE20K. The larger the marker, the more GMACs the model consumes. "SA" and "LA" indicate that the methods are based on Self-Attention and Linear Attention. We measure the latency on iPhone13 with the batch size set as 1 and the 512×512 input resolution. We build the segmentation head of our models using Semantic FPN [24].

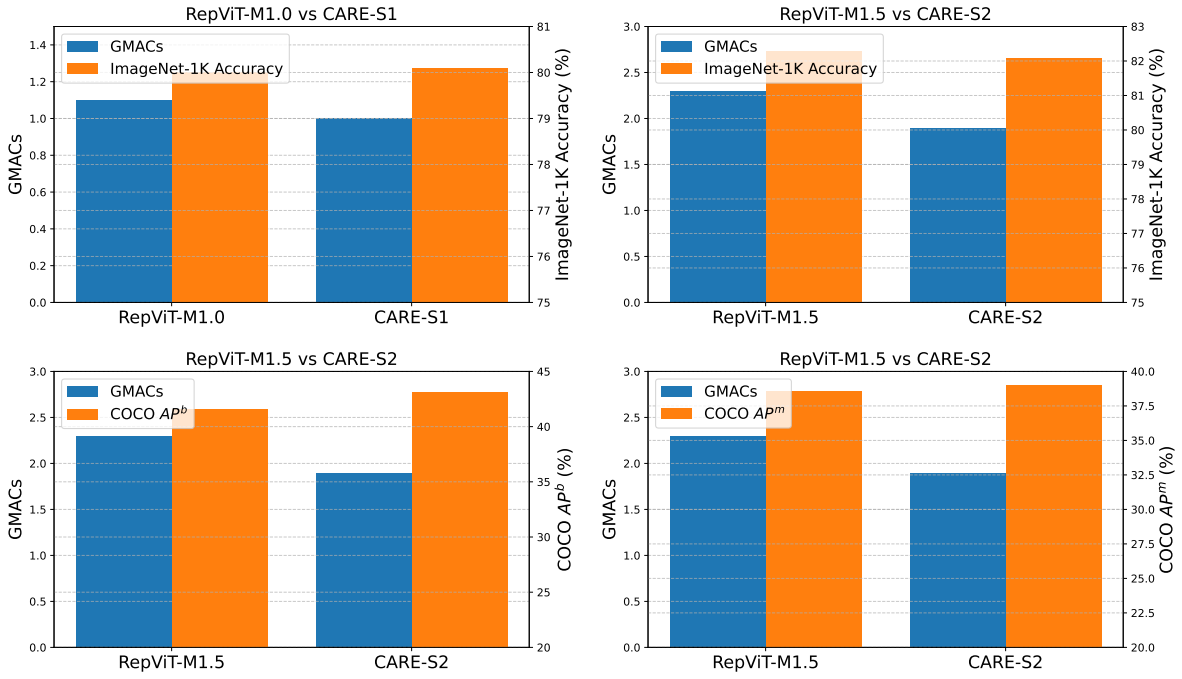


Figure 9. The visualized comparison between RepViT [46] and our CARE Transformers in terms of accuracy and efficiency on COCO and ImageNet-1K datasets.

S2 is slightly lower than that of Agent-Swin-T and FLatten-Swin-T (FLatten-Swin-T and Agent-Swin-T are the largest-size models of FLatten and Agent presented in Figure 8), it has obviously better efficiency, *e.g.,* the speed of CARE-S2 is about $3\times$ faster than that of Agent-Swin-T and FLatten-Swin-T. These results indicate that our CARE-Transformers are more suitable to be deployed in resource-constrained scenarios.

8.4. MLLA V.S. CARE

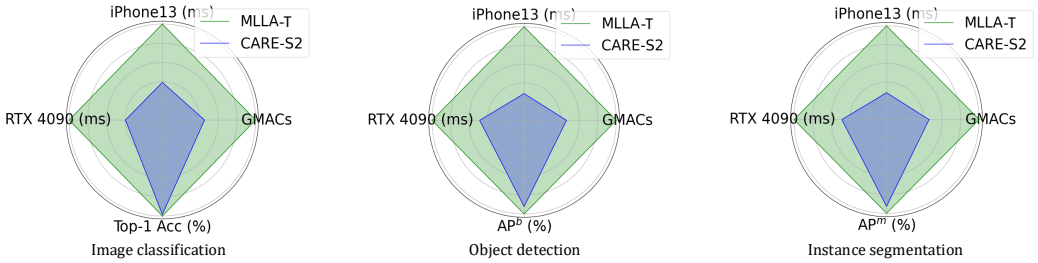
Our method can be seen as playing a step further on MLLA [16], and therefore we provide more quantitative comparisons between MLLA and our CARE Transformers, which are summarized in Figure 10. From the figure, we can see that our CARE-S2 has obviously better efficiency than both MLLA-T and MLLA-B but still maintains comparable accuracy on all of the image classification, object detection, instance segmentation, and semantic segmentation task. For example, on the image classification task of ImageNet-1K, our CARE-S2 just costs half the GMACs of MLLA-T and has $2.5\times$ faster speed on iPhone13, but its top-1 accuracy is only 1.4% lower than that of MLLA-T. Meanwhile, CARE-S2 requires only one-eighth the GMACs of MLLA-B, while its top-1 accuracy is just 3.2% lower than that of MLLA-B. Similar comparison results can be found on the other tasks as well. For example, on the instance segmentation task of COCO, the AP^m of CARE-S2 is lower than that of MLLA-T by 3.1% and MLLA-B by 6%, but it has about $3\times$ and $6\times$ faster execution speed on the mobile device iPhone13. These results indicate that our CARE Transformers are more suitable to be deployed in resource-constrained scenarios, and demonstrate our decoupled learning approach can achieve a better balance between accuracy and efficiency.

8.5. RepViT V.S. CARE

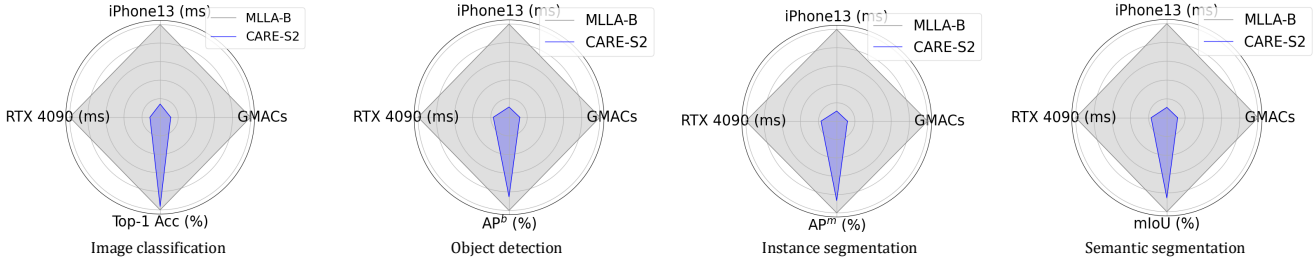
We also compare our method with a recent convolutional model RepViT [46], which is exhibited in Figure 9. The experimental results shown in Figure 9 indicate that our CARE-S1 can achieve higher accuracy than RepViT-M1.0 on the classification task of ImageNet-1K but still consumes less GMACs. Meanwhile, our CARE-S2 can achieve comparable classification accuracy to RepViT-M1.5 but has obviously higher efficiency, *e.g.*, CARE-S2 just needs to cost 80% of the GMACs of RepViT-M1.5. Similar results can also be found on COCO. These results demonstrate the competitive performance achieved by our CARE Transformers.

9. Visualization of Training Loss

In Figure 11, we also visualize the convergence of the loss during training our CARE Transformers on the ImageNet-1K dataset. The results in the figure indicate that the training loss of our CARE-S0, CARE-S1, and CARE-S2 con-



(a) MLLA-T V.S. CARE-S2



(b) MLLA-B V.S. CARE-S2

Figure 10. The detailed comparisons between MLLA [16] and our CARE Transformers. Following [16], we build the detection head by using Mask R-CNN [21] and utilize UperNet [53] to implement the segmentation head of our CARE Transformers. As [16] only applies MLLA-B to the semantic segmentation task, we lack the comparisons between MLLA-T and CARE-S2 on the ADE20K dataset.

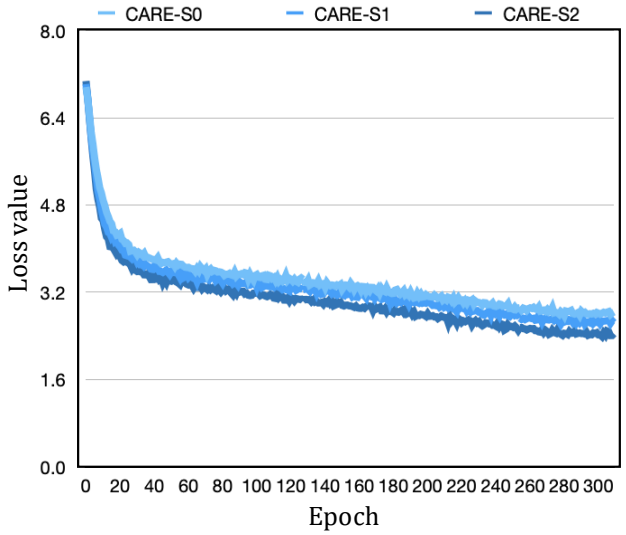


Figure 11. Visualization for the convergence of the loss during training our CARE-S0, CARE-S1, and CARE-S2 models on ImageNet-1K.

verges to a relatively small value within only a few training epochs (about 20 training epochs), indicating that our CARE Transformers based on the asymmetrical decoupled learning strategy can be easily optimized by using back-propagation and stochastic gradient descent algorithms.

10. Additional Proofs

∋ **Proof 2.** *When learning local inductive bias in multi-scale receptive field using Equation (14), (15), and (16), our ∋* ***Proposition 3*** *(in Section 3.3 of the main paper) can still hold.*

We can rewrite Equation 9 of the main paper to the multiscale version as follows:

$$\Omega(\Delta) = \frac{1}{2}hw(k_1^2 + k_2^2)(d + \Delta) + \frac{3}{2}hw(d - \Delta)^2. \quad (17)$$

Therefore,

$$\Omega(\Delta_1) - \Omega(\Delta_2) \quad (18)$$

$$= \frac{3}{2}hw(\Delta_1 - \Delta_2)(\frac{1}{3}k_1^2 + \frac{1}{3}k_2^2 + \Delta_1 + \Delta_2 - 2d). \quad (19)$$

Let $\Delta_1 > 0$ and $\Delta_2 = 0$, we have

$$\Omega(\Delta_1) - \Omega(\Delta_2) = \frac{3}{2}hw\Delta_1(\frac{1}{3}k_1^2 + \frac{1}{3}k_2^2 + \Delta_1 - 2d). \quad (20)$$

As we set $k_1 = 3$ and $k_2 = 7$ in the paper, we can obtain,

$$\Omega(\Delta_1) - \Omega(\Delta_2) = \frac{3}{2}hw\Delta_1(3 + \frac{49}{3} + \Delta_1 - 2d) \quad (21)$$

$$< \frac{3}{2}hw\Delta_1(20 + \Delta_1 - 2d). \quad (22)$$

Despite that the dimension of features is set as a small value $d = 24$ in the first stage of our CARE Transformers, it still satisfies $\Omega(\Delta_1) - \Omega(\Delta_2) < 0$, as $\Delta_1 < d$ and $20 < d$, proving that the asymmetrical setting ($\Delta_1 > 0$) still has less complexity compared to the symmetrical scenario ($\Delta_2 = 0$). In most cases, the dimension of features is generally set to be larger than the value 20 in deep neural networks.

We extend ∋ **Proof 1.** (in Section 3.3 of the main paper) to the version of using multi-head linear attention.

∋ **Proof 3.** *With the use of multi-head linear attention, our asymmetrical setting $d_1 < d_2$ can still reduce computation complexity .*

We first rewrite Equation (8) into the multi-head version:

$$\Omega = 2k^2hwd_2 + 4hwd_1^2 + 2hw\frac{d_1^2}{n}, \tag{23}$$

where d_1 and d_2 denote the dimensions of local and global features, $k \times k$ indicates the kernel size of the convolutional local bias learner, and n represents the head number. Let $d_2 - d_1 = \Delta$, we have $d_1 = \frac{d-\Delta}{2}$ and $d_2 = \frac{d+\Delta}{2}$ due to $d_1 + d_2 = d$ and $d_2 - d_1 = \Delta$. We can rewrite Equation (23) as follows:

$$\Omega(\Delta) = k^2hw(d+\Delta) + (1+\frac{1}{2n})hw(d-\Delta)^2. \tag{24}$$

Accordingly, we have

$$\Omega(\Delta_1) - \Omega(\Delta_2) \tag{25}$$
$$= \frac{2n+1}{2n}hw(\Delta_1 - \Delta_2)(\frac{2n}{2n+1}k^2 + \Delta_1 + \Delta_2 - 2d) \tag{26}$$

Let $\Delta_1 > 0$ and $\Delta_2 = 0$, we also have $\Omega(\Delta_1) - \Omega(0) < 0$ as $\Delta_1 < d$ and the kernel size generally obeys $k^2 << d$, which proves that our asymmetrical setting ($\Delta_1 > 0$) has less complexity compared to the symmetrical scenario ($\Delta_2 = 0$).

We can also extend ∋ **Proof 2** to the version of multi-head linear attention.

∋ **Proof 4.** *When using Equation (14), (15), and (16) to learn local inductive bias in multiscale receptive field and employing multi-head linear attention at the same time, our ∋ **Proposition 3** (in Section 3.3 of the main paper) remains valid.*

We can rewrite Equation (17) of the supplementary material to the multi-head version, just like the process given in Equation (24):

$$\Omega(\Delta) = \frac{1}{2}hw(k_1^2 + k_2^2)(d+\Delta) + (1+\frac{1}{2n})hw(d-\Delta)^2. \tag{27}$$

Therefore,

$$\Omega(\Delta_1) - \Omega(\Delta_2) \tag{28}$$
$$= \frac{2n+1}{2n}hw(\Delta_1 - \Delta_2)[\frac{n(k_1^2+k_2^2)}{2n+1} + \Delta_1 + \Delta_2 - 2d]. \tag{29}$$

Let $\Delta_1 > 0$ and $\Delta_2 = 0$, we have

$$\Omega(\Delta_1) - \Omega(\Delta_2) = \frac{2n+1}{2n}hw\Delta_1[\frac{n(k_1^2+k_2^2)}{2n+1} + \Delta_1 - 2d] \tag{30}$$
$$< \frac{2n+1}{2n}hw\Delta_1(\frac{k_1^2+k_2^2}{2} + \Delta_1 - 2d) \tag{31}$$

As we set $k_1 = 3$ and $k_2 = 7$, and adopt the asymmetrical setting $d_1 = \frac{1}{2}d_2$ (i.e., $d_1 = \frac{1}{3}d$, $d_2 = \frac{2}{3}d$, and $\Delta_1 = d_2 - d_1 = \frac{1}{3}d$), we can obtain,

$$\Omega(\Delta_1) - \Omega(\Delta_2) = \frac{2n+1}{2n}hw\Delta_1(\frac{9+49}{2} + \frac{1}{3}d - 2d) \tag{32}$$
$$= \frac{2n+1}{2n}hw\Delta_1(29 - \frac{5}{3}d) \tag{33}$$

In the first stage of our CARE Transformers, the feature dimension is set as a small value $d = 24$ but it still satisfies $\Omega(\Delta_1) - \Omega(\Delta_2) < 0$. As $d > 24$ in the subsequent deeper layers of our models, they obviously obey the condition $\Omega(\Delta_1) - \Omega(\Delta_2) < 0$, which proves that our proposed asymmetrical decoupled learning strategy ($\Delta_1 > 0$) still has less complexity compared to the symmetrical learning way ($\Delta_2 = 0$).

11. Limitation & Future Work

Our current work has two limitations. Firstly, we have not yet employed neural architecture search (NAS) to optimize the architectural design of our CARE Transformers. Secondly, due to our limited GPU resources, we do not apply our approach to models with relatively larger size; however, we believe that the proposed CARE mechanism can also perform well with large models, as its superiority lies in the more effective and reasonable use of local and global information. We plan to address these limitations in our future work. Recently, Segment Anything Model (SAM) [25] has shown impressive zero-shot transfer performance. However, its heavy computation costs still limit its application in resource-constrained scenarios. In the future, we plan to incorporate SAM with our CARE mechanism and utilize our efficient CARE block to build the image encoder of SAM, making it possible to be deployed to resource-constrained devices.