International Journal of Computer Vision (2023) 131:2516–2528
https://doi.org/10.1007/s11263-023-01813-x

MANUSCRIPT

What Limits the Performance of Local Self-attention?

Jingkai Zhou[1] · Pichao Wang[2] · Jiasheng Tang[3] · Fan Wang[4] · Qiong Liu[1] · Hao Li[3] · Rong Jin[2]

Received: 12 June 2022 / Accepted: 5 May 2023 / Published online: 9 June 2023

Abstract
Although self-attention is powerful in modeling long-range dependencies, the performance of local self-attention (LSA) is just similar to depth-wise convolution, which puzzles researchers on whether to use LSA or its counterparts, which one is better, and what limits the performance of LSA. To clarify these, we comprehensively investigate LSA and its counterparts from *channel setting* and *spatial processing*. We find that the devil lies in attention generation and application, where relative position embedding and neighboring filter application are key factors. Based on these findings, we propose enhanced local self-attention (ELSA) with Hadamard attention and the ghost head. Hadamard attention introduces the Hadamard product to efficiently generate attention in the neighboring area, while maintaining the high-order mapping. The ghost head combines attention maps with static matrices to increase channel capacity. Experiments demonstrate the effectiveness of ELSA. Without architecture/hyperparameter modification, drop-in replacing LSA with ELSA boosts Swin Transformer by up to +1.4 on top-1 accuracy. ELSA also consistently benefits VOLO from D1 to D5, where ELSA-VOLO-D5 achieves 87.2 on the ImageNet-1K without extra training images. In addition, we evaluate ELSA in downstream tasks. ELSA significantly improves the baseline by up to +1.9 box Ap/+1.3 mask Ap on the COCO, and by up to +1.9 mIoU on the ADE20K.

Keywords Vision transformer · Local self-attention · Representation learning · Image classification

1 Introduction

From upstream to downstream visual tasks, vision transformers (Dosovitskiy et al., 2020; Touvron et al., 2021; Carion et al., 2020; Zheng et al., 2021; Sun et al., 2020; He et al., 2021) have set off a revolution by achieving promising results. Behind the success, the multi-head self-attention (MHSA) plays a critical role, which generates attention maps to dynamically aggregate spatial information, leading to greater flexibility and larger capacity. Recent studies (Raghu et al., 2021; Yuan et al., 2021a) demonstrated that MHSA tends to focus on local information in the first few layers of vision transformers. Several methods (Dai et al., 2021; Chen et al., 2021d; Yuan et al., 2022; Zhang et al., 2021) introduce inductive bias to force earlier layers to embed local details, which boosts the generalization ability of vision transformers. As a representative of them, Swin Transformer (Liu et al., 2021) brings locality to MHSA and makes great progress on a wide range of visual tasks.

However, one strange phenomenon appears in Swin Transformer. One can achieve similar performance when replacing local self-attention (LSA) in Swin Transformer with depth-wise convolution (DwConv) (Chollet, 2017; Howard et al.,

Work done during an internship at Alibaba Group.

✉ Pichao Wang
pichaowang@gmail.com

Jingkai Zhou
fs.jingkaizhou@gmail.com

Jiasheng Tang
jiasheng.tjs@alibaba-inc.com

Fan Wang
fan.w@alibaba-inc.com

Qiong Liu
liuqiong@scut.edu.cn

Hao Li
lihao.lh@alibaba-inc.com

Rong Jin
jinrong.jr.w@alibaba-inc.com

1 South China University of Technology, Gruangzhou, Guangdong, China

2 Alibaba Group, Bellevue, WA, USA

3 Alibaba Group, Hangzhou, Zhejiang, China

4 Alibaba Group, Sunnyvale, CA, USA

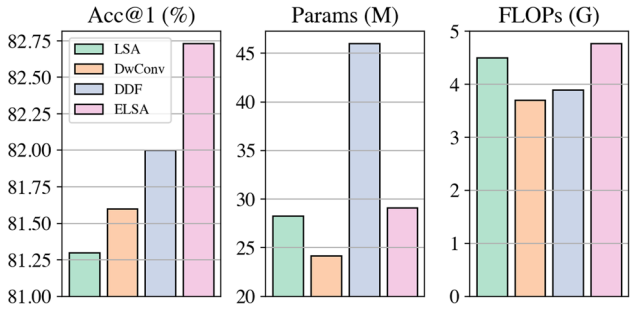

Fig. 1 Comparison of different layers on the Swin-T (Liu et al., 2021) architecture. The performance of local self-attention (LSA) is just similar to depth-wise convolution (DwConv), and inferior to dynamic filters, like DDF (Zhou et al., 2021b). Our ELSA surpasses these counterparts by a large margin while using a similar number of parameters and FLOPs as LSA

2017) or dynamic filters. As shown in Fig. 1, we replace LSA in Swin-T (Liu et al., 2021) with DwConv and the decoupled dynamic filter (DDF) (Zhou et al., 2021b). LSA only achieves similar top-1 accuracy as DwConv, which is lower than DDF, but it requires more floating-point operations (FLOPs). This phenomenon has also been observed in downstream COCO object detection and ADE semantic segmentation tasks by recent papers (Han et al., 2022; Yuan et al., 2022; Dai et al., 2021; Fang et al., 2021), which motivates us to raise a question: *what limits the performance of local self-attention?*

To answer this question, we thoroughly review LSA, DwConv, and dynamic filters from two key aspects: *channel setting* and *spatial processing*.

Channel Setting. One straightforward difference between DwConv and LSA is the channel setting. DwConv applies different filters to different channels. LSA adopts the multi-head strategy, which shares filters (a.k.a attention maps) within each group of channels. In this work, we consider the channel setting of DwConv as a special case of the multi-head strategy, where the number of heads is set to the number of channels. One guess is that more heads in DwConv might be a critical factor in why it performs comparably to LSA. However, our experiments show that even if we set the head number of DwConv to be the same as that of LSA, DwConv still achieves similar or better accuracy. We also find that directly increasing the head number of LSA will not improve its performance.

Spatial Processing. How to obtain and apply filters (or attention maps) to gather spatial information is another difference between DwConv, dynamic filters, and LSA. DwConv shares static filters across all feature pixels in a sliding window way. Dynamic filters (Wang et al., 2019; Li et al., 2021a; Yuan et al., 2022; Jia et al., 2016; Zhou et al., 2021b) employ a bypath network, normally a 1×1 convolution, to generate spatial-specific filters, and apply these filters to the neighboring area of each pixel. LSA generates attention maps, which are also a kind of spatial-specific filter, via the dot product of the query and key matrices. LSA applies these attention maps to local windows. In this work, we unify the above three kinds of spatial processing into one paradigm, and fairly investigate them from parameterization, normalization, and filter application. We find that the neighboring filter application and the parameterization strategy with relative position embedding are two key factors that affect performance.

Based on these findings, we propose the enhanced local self-attention module (ELSA) to better embed local information. ELSA employs relative position embedding in the parameterization strategy and applies generated attention (filters) to neighboring areas. Beyond these, ELSA further introduces Hadamard attention and the ghost head module. In Hadamard attention, we replace the dot product between queries and keys with the Hadamard product, which is more computational-friendly in neighboring areas while maintaining comparable performance. The ghost head combines a dynamic attention map with static ghost head filters to improve channel capacity. We empirically validate the performance of ELSA by drop-in replacing LSA / Outlooker (Yuan et al., 2022) in Swin Transformer (Liu et al., 2021) / VOLO (Yuan et al., 2022). Without changing the architecture / hyperparameter of other parts, ELSA considerably improves the performance of Swin Transformer (+1.4 Top-1 Acc) and VOLO (+0.7 Top-1 Acc), while introducing few parameters and FLOPs. In addition, we also demonstrate the superior performance of ELSA in downstream object detection (up to +1.9 box AP/+1.3 mask AP) and semantic segmentation tasks (up to +1.9 mIoU).

In short, we make the following contributions in this work:

- Extensively investigate DwConv, dynamic filters, and LSA to empirically reveal *which factors limit the performance of LSA*.
- Unify the special processing of DwConv, dynamic filters, and LSA into one paradigm so that they can be treated as the same thing with different settings.
- Propose the enhanced local self-attention (ELSA) to better embed local details by introducing Hadamard attention and the ghost head.
- Validate ELSA in both upstream and downstream tasks. The use of ELSA in drop-in replacement significantly and consistently boosts baseline methods.

2 Related Work

Vision Transformers Transformer (Vaswani et al., 2017) is first proposed in the NLP task and achieves dominant performance (Devlin et al., 2019; Brown et al., 2020). Recently, the pioneering work ViT (Dosovitskiy et al., 2020) successfully applies the pure transformer-based architecture to computer

vision, revealing the potential of transformer in handling visual tasks. Lots of follow-up studies are proposed (Graham et al., 2021; Guo et al., 2022; El-Nouby et al., 2021; Chen et al., 2021a; Huang et al., 2021; Mehta & Rastegari, 2021; Lu et al., 2021; Gu et al., 2021; Chen et al., 2022, 2021e; Li et al., 2021b; Dong et al., 2021; Yuan et al., 2021b). Many of them analyze the ViT (Wu et al., 2021; Li et al., 2021c; Gong et al., 2021; Chu et al., 2021; Dai et al., 2021; Xiao et al., 2021; Han et al., 2022; Yuan et al., 2021a; Wang et al., 2022b; Raghu et al., 2021; Pan et al., 2022) and improve it via introducing locality to earlier layers (Dai et al., 2021; Yuan et al., 2022; Vaswani et al., 2021; Chen et al., 2021d; Zhang et al., 2021; Yang et al., 2021; Liu et al., 2021). In particular, Raghu et al (Raghu et al., 2021) observe that the first few layers in ViTs focus on local information. Li et al (Yuan et al., 2021a) also demonstrate that the first few layers embed local details. Xiao et al (Xiao et al., 2021) and Wang et al (Wang et al., 2022a) find that introducing inductive bias, like convolution stem, can stabilize the training and improve the peak performance of ViTs. Similarly, Dai et al (Dai et al., 2021) marry convolution with ViTs, improving the model generalization ability. Swin Transformer (Liu et al., 2021, 2022a), as a milestone, also leverages local self-attention (LSA) to embed detailed information in high-resolution finer-level features. Despite these successes, several studies (Han et al., 2022; Dai et al., 2021; Yuan et al., 2022; Fang et al., 2021) observe that the performance of LSA is just on par with depth-wise convolution (Han et al., 2022; Liu et al., 2022b). The reasons behind this phenomenon are not clear, and in-depth comparisons under the same conditions are valuable.

Dynamic Filters. Convolution and depth-wise convolution (Chollet, 2017) has been widely used in CNNs (He et al., 2016; Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019; Tan & Le, 2019), while their content-agnostic nature limits the model flexibility and capacity. To solve this problem, dynamic filters are proposed one after another. One kind of dynamic filter (Ma et al., 2020; Yang et al., 2019; Zhang et al., 2020; Chen et al., 2020) predicts coefficients to combine several expert filters which are then shared across all spatial pixels. Another kind of dynamic filter (Jia et al., 2016; Chen et al., 2021c; Wang et al., 2019, 2021a; Li et al., 2021a; Su et al., 2019; Zhou et al., 2021b; Yuan et al., 2022) generates spatial-specific filters. Specifically, the dynamic filter networks (Jia et al., 2016) use the separate network branches to predict a complete filter at each pixel. PAC (Su et al., 2019) uses a fixed Gaussian kernel on adapting features to modify the standard convolution filter at each pixel. DRConv (Chen et al., 2021c) extends CondConv (Yang et al., 2019) to each pixel. CARAFE (Wang et al., 2019) and CARAFE++ (Wang et al., 2021a) are the dynamic layers for upsampling and downsampling, where a channel-shared 2D filter is predicted at each pixel. Similarly, Involution (Li et al., 2021a) applies the CARAFE-like structure to feature extraction. VOLO (Yuan et al., 2022) introduces the Outlooker to embed local details. DDF (Zhou et al., 2021b) decouples dynamic filters to spatial and channel ones, reducing computational overhead while achieving promising results. In this work, we observe that dynamic filters, like DDF (Zhou et al., 2021b), perform superior to LSA. Based on comparison and discussion, we empirically reveal the factors leading to such a phenomenon, and propose enhanced local self-attention (ELSA) to better embed local details.

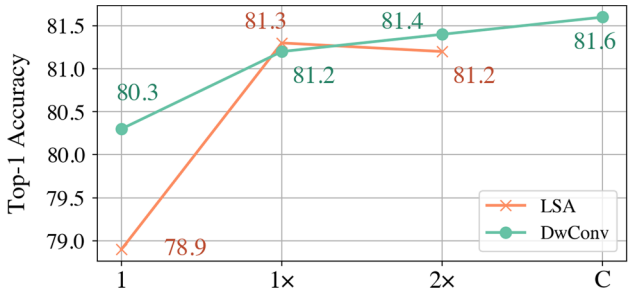

Fig. 2 Investigation of different channel settings. Only the number of heads is changed each time for fair comparisons. The LSA version runs out of memory under setting C

3 Channel Setting

To figure out *what limits the performance of LSA*, we first focus on one of the most obvious differences between DwConv and LSA, i.e, the channel setting. DwConv applies a static filter to each channel. Differently, several dynamic filters (Wang et al., 2019; Li et al., 2021a; Yuan et al., 2022) and LSA employ the multi-head strategy, which splits channels into multiple groups and shares the same filter within each group. In this work, we consider the setting of DwConv as a special case of the multi-head strategy, where the number of heads is equal to the number of channels. Therefore, comparing channel settings between LSA and DwConv is essentially investigating performance on different numbers of heads.

Figure 2 shows the investigation results on the ImageNet-1K dataset. We compare two versions of Swin-T (Liu et al., 2021) under the same hyperparameters, and only modify the head setting each time. Setting 1 means that the number of heads is set to 1 for all layers. Setting 1× represents the original setting of Swin-T, that is, the head numbers of four stages are set to {3, 6, 12, 24}. Setting 2× means to double the original setting, i.e, {6, 12, 24, 48}. Setting C denotes that the number of heads is set to the number of channels for all layers. There has been a guess that more heads in the DwConv version increase model capacity, thus may lead to comparable performance. However, we find that under the same

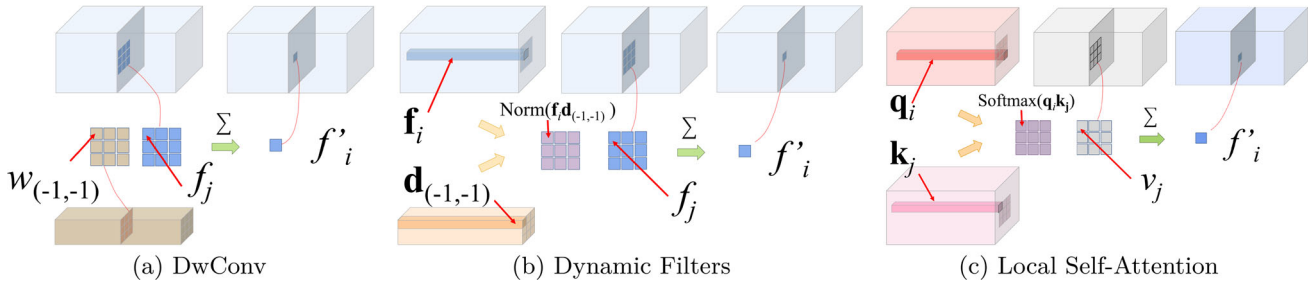

Fig. 3 Illustration of spatial processing strategies. DwConv applies the static convolutional filters to the neighboring area of each pixel. Dynamic filters generate dynamic weights and apply them to the neighboring area of each pixel. Local self-attention uses the query and key to generate a local attention map and applies it to the local window

channel setting, like $1\times$ and $2\times$, the DwConv version still achieves similar performance as the LSA one. The DwConv version even exceeds the LSA one under setting 1, which demonstrates that the channel setting is not the essential factor leading to the strange phenomenon.

In addition, we observe that directly setting more heads than $1\times$ does not benefit the LSA version. One possible reason is that more heads lead to fewer channels for each head generation, which compromises the quality of each head. As increasing the head number of static filters (DwConv) does benefit model performance, combining LSA with static filters is a promising idea to let LSA get benefit from more heads.

4 Spatial Processing

As the channel setting is not the critical factor, we seek the answer from the perspective of spatial processing. DwConv, dynamic filters, and LSA adopt different strategies to gather spatial information. We first review these strategies and unify them into one paradigm. Then, we fairly compare these strategies from three aspects.

4.1 Formulation

DwConv. DwConv does not generate filters. It shares the static convolutional filters by sliding windows. For a given channel, the spatial processing of DwConv can be written as

$$f'_i = \sum_{j\in\Theta} w_{j-i} f_j \tag{1}$$

where $f'_i \in \mathbb{R}$ represents the output feature value at pixel i, $f_j \in \mathbb{R}$ represents the input feature value at pixel j, $w_{j-i} \in \mathbb{R}$ is the filter weight with respect to the relative offset $j-i$. Take the filter size 3 as an example, $j-i$ corresponds to $\{(-1,-1), (-1,0), (-1,1), ..., (1,1)\}$. Θ notes the neighboring area around the pixel i. See Fig. 3a for illustration.

Dynamic Filters. Dynamic filters (Zhou et al., 2021b; Yuan et al., 2022; Li et al., 2021a; Wang et al., 2019) generate spatial-specific filters at each pixel via a bypath network and apply them to the neighboring area of that pixel. For a given channel, the spatial processing of dynamic filters can be written as

$$f'_i = \sum_{j\in\Theta} \mathrm{Norm}_\Theta(\mathbf{f}_i\mathbf{d}_{j-i}) f_j \tag{2}$$

where $f'_i \in \mathbb{R}$ represents the output feature value at pixel i, $f_j \in \mathbb{R}$ represents the input feature value at pixel j, $\mathbf{f}_i \in \mathbb{R}^c$ is the c-dimensional input feature vector at pixel i, $\mathbf{d}_{j-i} \in \mathbb{R}^c$ is the filter generation weight with respect to the relative offset $j-i$. Norm_Θ represents the normalization method applied to the generated filters, which can be the identity mapping in Involution (Li et al., 2021a), the filter normalization in DDF (Zhou et al., 2021b), or the softmax function in Outlooker (Yuan et al., 2022). Θ notes the neighboring area around the pixel i. Here, we omit the multi-head strategy for simplicity. Figure 3(b) illustrates this spatial processing strategy.

LSA. Unlike dynamic filters, LSA uses attention maps of local windows as spatial-specific filters. For a given channel, the spatial processing of LSA can be written as

$$f'_i = \sum_{j\in\Omega} \mathrm{Softmax}_\Omega(\mathbf{q}_i\mathbf{k}_j) v_j \tag{3}$$

where $f'_i \in \mathbb{R}$ represents the output feature value at pixel i, $\mathbf{q}_i, \mathbf{k}_j \in \mathbb{R}^c$ are the c-dimensional query / key vectors at pixel i and j, respectively. $v_j \in \mathbb{R}$ is the value scalar at pixel j. Query/key/value are generated from the input feature via linear mappings. Ω notes the local window area. $\mathrm{Softmax}_\Omega$ represents the softmax function applied to the generated attention among local window Ω. Here, we omit the multi-head strategy for simplicity. Figure 3c shows the spatial processing of LSA.

Unified Paradigm. To unify the above spatial strategies, we first consider $\mathbf{d}_{j-i}$ in Eq. (2) as a kind of relative position embedding, and consider w_{j-i} in Eq. (1) as a kind of relative position bias. Then, these spatial processing strategies can be unified into one paradigm, which can be written as

$$f_i' = \sum_{j \in \Phi} \mathrm{Norm}_\Phi(\mathbf{q}_i \mathbf{k}_j + \mathbf{q}_i \mathbf{r}^k_{j-i} + \mathbf{r}^q_{j-i} \mathbf{k}_j + r^b_{j-i}) v_j \quad (4)$$

where Φ can be either the local window Ω or the neighboring area Θ; Norm_Φ can be either the identity mapping, the filter normalization, or softmax; $\mathbf{r}^k_{j-i}, \mathbf{r}^q_{j-i} \in \mathbb{R}^c$ are relative position embeddings, and $r^b_{j-i} \in \mathbb{R}$ denotes the relative position bias.

DwConv, dynamic filters, and LSA are all special cases of this unified paradigm. For example, when only using r^b_{j-i} as parameterization, leveraging the identity mapping as Norm_Φ, and adopting the neighboring area Θ as Φ, Eq. (4) will degenerate to DwConv. Similarly, if the parameterization is set to $\mathbf{q}_i \mathbf{r}^k_{j-i}$, Norm_Φ is set to the identity mapping, and Φ is set to the neighboring area Θ, then Eq. (4) becomes a variant of Involution, where $\mathbf{r}^k_{j-i}$ is equivalent to $\mathbf{d}_{j-i}$ in Eq. (2). We can also get LSA from this paradigm by changing the parameterization, Norm_Φ, and Φ. Therefore, the spatial processing of LSA and its counterparts are essentially different in three factors: *parameterization*, *normalization*, and *filter application*. We then investigate each factor through extensive empirical studies.

4.2 Investigation

Parameterization. We first compare the effect of different parameterizations by setting Φ as the local window and setting Norm_Φ as softmax. The results are exhibited in Table 1. As can be seen, when applying filters to local windows, the parameterization strategy of dynamic filters (Net2) is better than that of the standard LSA (Net1). Also, a variant of dynamic filters (Net6) is on par with the LSA in Swin-T. Moreover, Net7 indicates that combining the parameterization of LSA with dynamic filters can further boost performance.

Normalization. Normalization is another factor that may influence the performance. DwConv and Involution (Li et al., 2021a) adopt identity mapping as normalization. DDF (Zhou et al., 2021b) introduces filter normalization. LSA and Outlooker (Yuan et al., 2022) employ the softmax function as normalization. We fairly compare these options under the same conditions. We choose the Net7 in Table 1 as the baseline, and only change the normalization part each time. As can be seen in Table 2, the identity mapping causes the training crash, and softmax is better than the filter normalization. This indicates that the normalization part should not be blamed for the limited performance of LSA.

Table 1 Comparison of different parameterization

Model	$\mathbf{q}_i \mathbf{k}_j$	$\mathbf{q}_i \mathbf{r}^k_{j-i}$	$\mathbf{r}^q_{j-i} \mathbf{k}_j$	r^b_{j-i}	Acc@1
Swin-T	✓			✓	81.3
Net1	✓				80.1
Net2		✓			80.9
Net3			✓		80.9
Net4				✓	79.8
Net5		✓	✓		81.1
Net6		✓	✓	✓	81.3
Net7	✓	✓	✓	✓	81.8

Swin-T is chosen as the baseline, models are trained under the same protocol. Acc@1 means the top-1 accuracy

Table 2 Comparison of different normalization

Model	Identity	Filter norm	Softmax	Acc@1
Net7	✓			Crash
		✓		81.4
			✓	81.8

We only switch the normalization of the Net7 in for fair comparison

Filter Application. The last factor of spatial processing is how filters are applied. LSA in Swin Transformer applies attention maps to non-overlapping local windows. In contrast, DwConv and dynamic filters apply filters to sliding neighboring areas. This difference can be described as using a different setting of Φ in Eq. (4). We choose Swin-T, Net6, and Net7 as the baseline, and switch Φ in the first three stages of those models. We implement the neighboring case Net7 (noted as Net7-N) in two ways. The first way uses SDC (Zhang et al., 2021) to save GPU memory, so that we can train Net7-N on one node (8× V100) like other models. The second way uses the unfold operation, which consumes huge GPU memories and requires two nodes (16× V100) to train. Table 3 shows the comparison results. When applying filters to neighboring areas, both Net6 and Net7 get significantly improved, indicating that the neighboring application is critical for the final performance.

4.3 Discussion

Key Factors. Based on the above investigations, the factors that limit LSA can be summarized in two folds: one key factor affecting performance is relative position embeddings. Net5 and Net6 with relative position embedding $\mathbf{r}^k_{j-i}$ and $\mathbf{r}^q_{j-i}$ achieve similar performance as Swin-T. Net7 further surpasses Swin-T by 0.5% on top-1 accuracy. The other critical factor is the filter application. Applying filters on

Table 3 Comparison of different filter applications

Model	Local window	Neighboring	Acc@1
Swin-T	✓		81.3
		✓	81.4
Net6	✓		81.3
		✓	82.0
Net7	✓		81.8
		✓	82.0†
		✓	82.4‡

We only switch the setting of the Φ in Eq. (4) for a fair comparison. †Denotes that the model is implemented using SDC and trained on a single node. ‡ denotes that the model is implemented using the unfold operation and trained on multiple nodes

query-centered neighboring areas considerably boosts the performance of Net6 and Net7. So far, we can empirically answer the question we raised at the beginning. The reason why DwConv can match the performance of LSA is because of the neighboring filter application. Without that, DwConv degenerates to a variant similar to Net4, which is inferior to LSA. Similarly, the reasons why DDF performs better than LSA are because of the relative position embedding and the neighboring filter application. Integrating these factors into LSA (Net7-N) achieves the best performance among all these variants.

Local Window v.s. Neighboring. The peak performance of the local window version is worse than the neighboring one. One possible reason is that the local window with window shifting may not be sufficient for information exchange. Another disadvantage of the local window is that it limits the macro-architecture design. One has to set pairs of layers at each stage to implement window shifting.

There is no such thing as a free lunch. The drawback of the neighboring version is low throughput. It is not easy to calculate the dot product between quires and keys in sliding neighboring areas. It requires sliding chunk (Zhang et al., 2021), unfold operations, or specialized CUDA implementations, which are either memory-consuming or time-consuming. How to avoid the dot product while maintaining good performance is a challenging problem.

Comparison with Concurrent Work. (Han et al. 2022) also observe that LSA achieves similar performance to DwConv in Swin Transformer and conduct extensive experiments to discuss this interesting phenomenon. However, this paper differs from them in the method unification and conclusion.

(Han et al. 2022) unify LSA and DwConv in terms of sparse connectivity, weight sharing, and dynamic weight. Instead of general summarizing, we first unify three operations into one equation, then split each different factor one by one, which provides more detailed information and, more importantly, comes to a different conclusion.

(Han et al. 2022) claim that weight sharing across positions is the key reason that boosts the performance of DwConv. Instead, we observe that **the key reason for improving the performance of DwConv is the filter application rather than weight sharing.** See Net4 in Table 1, the weights of Net4 are parameterized by relative bias r^b_{j-i}, which are shared across positions. Even so, the performance of Net4 is poor because the filters are applied to the local window.

Note that Net4 is different from Local MLPs, but similar to the local window version of DwConv. Local MLPs do not share weights between internal pixels of a local window. Instead, Net4 uses the same weight for the internal pixels of a local window. In other words, the weights of Local MLP are window-wise shared, while the weights of Net4 are position-wise shared. From the perspective of weight sharing, Net4 and DwConv are the same. The main difference between Net4 and DwConv is the filter application. The weights of Net4 are applied to local windows instead of neighboring areas, which limits the performance of Net4.

Also, see Net6 and Net7 in Table 3. The weight-sharing strategies of these two models are unchanged between the local window and neighboring versions. The only difference lies in the filter application area, which affects performance significantly.

5 Enhanced Local Self-Attention

In addition to answering the question we raised, more importantly, we design a new local self-attention module that surpasses both the LSA in Swin Transformer and dynamic filters. This is accomplished with our enhanced local self-attention (ELSA), where the key techniques are Hadamard attention and the ghost head module. For a given channel, ELSA can be written as

$$f'_i = \sum_{j \in \Theta} \mathrm{G}(h_{j-i}) v_j \tag{5}$$

where h_{j-i} is the Hadamard attention value, $\mathrm{G}(\cdot)$ notes the ghost head mapping, f'_i is the output feature at pixel i, and v_j is the value scalar at pixel j. Figure 4 illustrates the overall structure of ELSA.

Hadamard Attention. First, we review Net7-N and Net6-N, which get the highest and the second-highest accuracy in our investigation. As discussed above, Net7-N brings in difficulties on implementation and inference. In the neighboring case, $\mathbf{q}_i \mathbf{k}_j$ in Net7-N needs to be implemented using unfold, sliding chuncks (Zhang et al., 2021), or CUDA operations, which are either memory-consuming or time-consuming. As

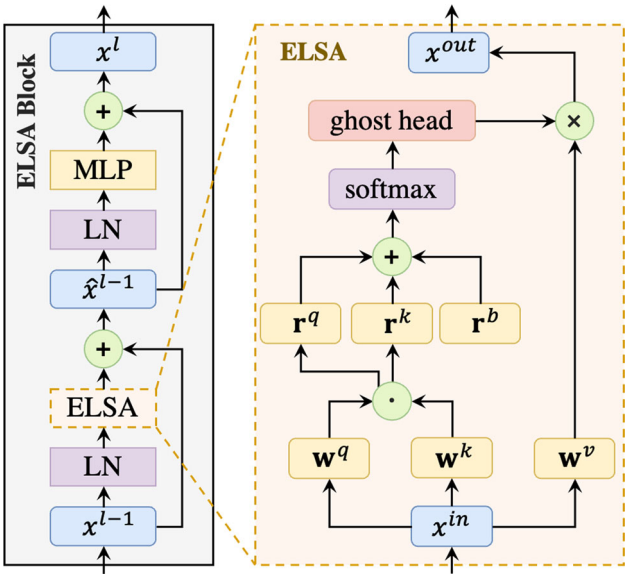

Fig. 4 Illustration of the ELSA block. ELSA can seamlessly replace LSA or dynamic filters in models

shown in Eq. 6, Net6-N removes the dot product term, thus getting rid of difficulties in filter / attention generation.

$$f_i' = \sum_{j \in \Theta} \text{Softmax}_\Theta(\mathbf{q}_i \mathbf{r}^k_{j-i} + \mathbf{r}^q_{j-i} \mathbf{k}_j + r^b_{j-i}) v_j \tag{6}$$

However, the performance of Net6-N is slightly worse than Net7-N. We find that the lower accuracy of Net6-N may be due to the lower mapping order. Specifically, Net-7 can be considered as a third-order mapping of the input, because Net-7 contains the second-order term $\mathbf{q}_i \mathbf{k}_j$ and combine it with the value $\mathbf{v}$. However, since Net6-N removes the dot product $\mathbf{q}_i \mathbf{k}_j$, which becomes a second-order mapping of the input.

One common hypothesis in deep learning is that the mappings with the higher order have stronger fitting ability (Lin & Ye, 2016). Recent concurrent work HorNet (Rao et al., 2022) also reveals the importance of higher-order mappings. Thus, we want to design a module that maintains the third-order mapping just like Net7-N, but without using the dot product of the query and key matrices. To accomplish this, we propose Hadamard attention which introduces Hadamard product between queries and keys as the second-order term. In terms of formulation, Hadamard attention can be written as

$$h_{j-i} = \text{Softmax}_\Theta((\mathbf{q}_i \odot \mathbf{k}_i)\mathbf{r}^k_{j-i} + \mathbf{r}^q_{j-i}(\mathbf{q}_j \odot \mathbf{k}_j) + r^b_{j-i}) \tag{7}$$

where $\odot$ means Hadamard product. With this simple yet effective modification, Eq. (7) becomes a third-order representation of the input. It is worth noting that Hadamard attention encodes spatial correlation by addition rather than multiplication. See Eq. (7). $(\mathbf{q}_i \odot \mathbf{k}_i)\mathbf{r}^k_{j-i}$ comes from pixel i, $\mathbf{r}^q_{j-i}(\mathbf{q}_j \odot \mathbf{k}_j)$ comes from pixel j. These two terms are summed to collect the information at pixels i and j. Thus, even though there is no dot product term $\mathbf{q}_i \mathbf{k} j$ in the Hadamard attention, it is still able to embed the spatial correlation between pixel i and j, just like the standard self-attention.

In addition, Hadamard attention is easy to implement. Unlike Net7-N that needs memory-/time-consuming operations to calculate $\mathbf{q}_i \mathbf{k}_j$, $\mathbf{q}_i \odot \mathbf{k}_i$ and $\mathbf{q}_j \odot \mathbf{k}_j$ can be implemented via simple element-wise multiplication of the query and key feature maps. Also, $\mathbf{r}^k_{j-i}$ and $\mathbf{r}^q_{j-i}$ are equivalent to 1×1 convolutional filters. Thus, Eq. (7) can be implemented by feature multiplication followed by convolution layers.

Algorithm 1 Demo code of ghost head (PyTorch-like)

```
# B: batch size, C: channel size
# N: the number of pixels
# H: the number of heads, K: kernel size
# h_attn: Hadamard attention with size (B, H, N, K*K)
# lambda, gamma: hyperparameters

def init()
mul_matrix = nn.Parameters(torch.randn(C, K, K))
add_matrix = nn.Parameters(torch.zeros(C, K, K))
trunc_normal_(add_matrix, std=0.02)

def ghost_head(h_attn):
# change the size of h_attn to (B, 1, H, N, K*K)
h_attn = h_attn.unsqueeze(1)

# reshape the size of matrices
mul_matrix = mul_matrix.reshape(1, C//H, H, 1, K*K)
add_matrix = add_matrix.reshape(1, C//H, H, 1, K*K)

# combination
h_attn = (mul_matrix ** lambda) * h_attn + gamma *
     add_matrix
return h_attn.reshape(B, C, N, K*K)
```

Ghost Head. In Sect. 3, we reveal the relation between the number of heads and model performance. More heads than setting $1\times$ cannot improve LSA, but still slightly improve the performance of static filters (DwConv). Combining LSA with static filters is a promising way to let LSA get benefit from more heads. Based on this idea, we propose the ghost head module which expands the original heads by combining Hadamard attention with two static filters. The demo code of the ghost head is summarized in Algorithm 1. The 'mul_matrix' and 'add_matrix' are two learnable static matrices, the 'lambda' and 'gamma' are two hyperparameters. After combination, the heads of Hadamard attention get expanded. In the real implementation, we write CUDA operations to avoid large GPU memory consumption.

The ghost head is a cheap module. Its overhead is only $O(n_c \times ks \times ks \times n_p)$, where n_c is the number of channels, ks is the filter size (i.e the size of neighboring areas), and n_p is the number of pixels. Recently, Refiner (Zhou et al., 2021a) is also proposed to adjust attention after softmax. Unlike them, the ghost head does not leverage heavy convolutions and

linear mappings, but only uses two simple static matrices. Also, the main purpose of the ghost head is not to refine attention, but to enrich channel capacity.

6 Experiments

We evaluate our ELSA in the Swin Transformer and VOLO architectures on ImageNet-1K image classification (Deng et al., 2009), COCO object detection (Lin et al., 2014), and ADE20K semantic segmentation (Zhou et al., 2017). For Swin Transformer (Liu et al., 2021), we drop-in replace LSA with ELSA without changing any architecture / hyperparameters. For VOLO (Yuan et al., 2022), we adjust several hyperparameters of ELSA Blocks, while maintaining all other parts unchanged.

6.1 Image Classification on ImageNet-1K

Settings. Our major evaluations are conducted on the ImageNet-1K (Deng et al., 2009) dataset. During training, no extra training images are used. Our code is based on Pytorch (Paszke et al., 2019), timm (Wightman, 2019), DDF (Zhou et al., 2021b), Swin Transformer (Liu et al., 2021), and VOLO (Yuan et al., 2022). The detailed setups are as follows.

Swin Transformer. We replace the LSA blocks of the first three stages with our ELSA blocks, and train ELSA-Swin following (Liu et al., 2021). In particular, AdamW (Loshchilov & Hutter, 2017) is selected as the optimizer. The base learning rate is set to 1e-3, which is scaled following the linear strategy, i.e $lr = lr_{base} \times \frac{batch\_size}{1024}$, and decays following the cosine strategy. We train models for 310 epochs, where the first 20 epochs are used for warm-up, and the last 10 epochs are used for cool-down. The weight decay of 5e-2 is used. We also leverage the same augmentation and regularization strategies as (Liu et al., 2021). Exponential moving average (EMA) (Polyak & Juditsky, 1992; Tarvainen & Valpola, 2017) is used in ELSA-Swin-B training. All models are trained / evaluated on 224×224 resolution unless otherwise specified.

VOLO. We replace all Outlooker modules in VOLO with ELSA. We train ELSA-VOLOs following the training protocol of VOLO. Most settings of VOLO are similar to those of Swin Transformer, except for the following differences. The base learning rate is set to 1.6e−3 for VOLO-D1 and 8e-4 for VOLO-D5. The Token Labeling (Jiang et al., 2021) is used during training, thus, MixUp (Zhang et al., 2017) and CutMix (Yun et al., 2019) are replaced by MixToken (Jiang et al., 2021). EMA is used in ELSA-VOLO-D5 training. Please refer to (Yuan et al., 2022) for more details.

Table 4 Evaluate different components of ELSA

HA	GH	Params	FLOPs	Acc@1
Base Model		28.3M	4.5G	81.3
✓		29.0M	4.7G	82.4
✓	✓	29.1M	4.8G	**82.7**

The Swin-T is chosen as the baseline. HA means Hadamard attention, and GH denotes the ghost head module

Table 5 Comparison of models with different mapping order

Model	1st-order	2nd-order	3rd-order	Acc@1
DwConv-Swin-T	✓			81.6
Net6-N		✓		82.0
Net7-N			✓	82.4
ELSA w/o GH			✓	**82.4**

The compared models use Swin-T as the macro-architecture and apply filters to the neighboring area

Ablation study. We respectively analyze the effect of Hadamard attention and the ghost head module in ELSA. We choose Swin-T (Liu et al., 2021) as our base architecture and experiment with different modifications to ELSA. Table 4 shows the results of ablation experiments. The performance of Swin-T is improved by 1.1%, with only Hadamard attention. We observed another 0.3% improvement by plugging in the ghost head.

To support our hypothesis about the high-order mapping, we compared DwConv-Swin-T, Net6-N, Net7-N, and ELSA without ghost head (ELSA w/o GH) in Table 5. All of these models use Swin-T as the macro-architecture and apply filters to the neighboring area. The only difference between Net6-N (Eq. 6) and Net7-N (Eq. 4) is that Net6-N lacks the third-order term $\mathbf{q}_i\mathbf{k}_j$ in weight parameterization. The only difference between Net6-N (Eq. 6) and ELSA w/o GH (Eq. 7) is that $\mathbf{q}_i$ and $\mathbf{k}_j$ are replaced by $\mathbf{q}_i \odot \mathbf{k}_i$ and $\mathbf{q}_i \odot \mathbf{k}_i$ to perform the third-order mapping. As can be seen in Table 5, better performance is obtained as the mapping order is increased. By performing the third-order mapping, ELSA w/o GH obtained 0.4% higher top-1 accuracy than Net6-N. ELSA w/o GH achieves the same accuracy as Net7-N as they both perform the third-order mapping. Recent concurrent work HorNet (Rao et al., 2022) also uses higher-order mappings to improve the performance of pure CNNs. Compared with HorNet, ELSA applies dynamic kernels instead of static DwConv to provide dynamic spatial aggregation capability. Assessing the importance of this dynamic spatial aggregation is an interesting topic worthy of future study. In addition, it is expected to achieve a more powerful network by further increasing the order of ELSA using the structure of HorNet.

Table 6 further compares the performance between different types of layers on both Swin Transformer and VOLO. As

Table 6 Comparison of different layers

Architecture	Layer type	Params	FLOPs	Acc@1
Swin-T	LSA	28 M	4.5G	81.3
	DwConv	24 M	3.7G	81.6
	D-DwConv	51 M	3.8G	81.9
	DDF	46 M	3.9G	82.0
	ELSA	29 M	4.8G	**82.7**
VOLO-D1	LSA	27 M	–	83.8
	DwConv	27 M	–	83.8
	Outlooker	27 M	7.1G	84.2
	ELSA	27 M	8.0G	**84.7**

ELSA consistently boosts baselines and surpasses compared counterparts, while using overhead similar to LSA

Table 7 Compare throughput of layers

Layer Type	LSA	DwConv	Net7-N	ELSA
Throughput	712	893	167	531
Acc@1	81.3	81.6	82.4	82.7

The throughput is tested on a single V100 GPU with the input size $128 \times 3 \times 224 \times 224$

can be seen, Swin-T and VOLO-D1 with ELSA respectively achieve 82.7% and 84.7% top-1 accuracy, which surpass other compared counterparts. Note that on the very powerful baseline VOLO-D1, the Outlooker (Yuan et al., 2022) introduced in VOLO is only 0.4% higher than DwConv, the LSA obtains the same accuracy as DwConv, while our ELSA exceeds DwConv by 0.9%, which is non-trivial.

Table 7 compares the throughput between Swin-T (Liu et al., 2021), DwConv-Swin-T, Net7-N, and ELSA-Swin-T. As discussed before, $\mathbf{q}_i \mathbf{k}_j$ in Net7-N brings difficulties in implementation and inference. By replacing $\mathbf{q}_i \mathbf{k}_j$ with Hadamard product, the throughput of ELSA significantly surpasses that of Net7-N. Since the underlying implementation of the neighboring filter application is not as fully optimized as DwConv or matrix multiplication, ELSA is slower than LSA and DwConv. However, it can hopefully be improved by better bottom-level optimization and hardware improvement.

Compare with the state-of-the-art models. We compare ELSA-Swin and ELSA-VOLO with other state-of-the-art models in Table 8. For fair comparisons, results are split into groups according to the number of parameters.

As can be seen, for different model sizes, our proposed ELSA consistently boosts Swin Transformer and VOLO, while introducing little overhead. In particular, ELSA improves Swin-T, Swin-S, and Swin-B by 1.4%, 0.5%, and 0.5%, where ELSA-Swin-S is comparable to the original Swin-B with two-thirds of parameters and FLOPs. Testing on the resolution of 224, ELSA-VOLO-D1 and ELSA-VOLO-D3 yield 84.7% and 85.7% top-1 accuracy, respectively. The performance of ELSA-VOLO-D3 matches the performance of the original VOLO-D4. However, VOLO-D4 costs more than 2× parameters against ELSA-VOLO-D3. Without addi-

Table 8 Comparison of different backbones on the ImageNet-1K

Model	Params	FLOPs	#Res	Acc@1
T2T-ViT-14	22 M	5.2G	224^2	81.5
CoAtNet-0	25 M	4.2G	224^2	81.6
Twins-SVT-S	24 M	2.9G	224^2	81.7
LIT-S	21 M	4.1G	224^2	81.5
Swin-T	28 M	4.5G	224^2	81.3
SwinV2-T	28 M	5.9G	256^2	81.8
ConvNeXt-T	29 M	4.5G	224^2	82.1
VOLO-D1	27 M	7.1G	224^2	84.2
VOLO-D1$_{\uparrow 384}$	27 M	20.8G	384^2	85.2
ELSA-Swin-T	29 M	4.8G	224^2	**82.7**
ELSA-VOLO-D1	27 M	8.0G	224^2	**84.7**
ELSA-VOLO-D1$_{\uparrow 384}$	27 M	23.3G	384^2	**85.7**
T2T-ViT-24	64 M	15.0G	224^2	82.6
CoAtNet-1	42 M	8.4G	224^2	83.3
Twins-SVT-B	56 M	8.6G	224^2	83.2
LIT-M	48 M	8.6G	224^2	83.0
Swin-S	50 M	8.7G	224^2	83.0
SwinV2-S	50 M	11.5G	256^2	83.7
ConvNeXt-S	50 M	8.7G	224^2	83.1
ELSA-Swin-S	53 M	9.6G	224^2	**83.5**
CoAtNet-2	75 M	15.7G	224^2	84.1
Twins-SVT-L	99 M	15.1G	224^2	83.7
LIT-B	48 M	8.6G	224^2	83.4
Swin-B	88 M	15.4G	224^2	83.5
SwinV2-B	88 M	20.3G	256^2	84.2
ConvNeXt-B	89 M	15.4G	224^2	83.8
VOLO-D3	86 M	20.9G	224^2	85.4
VOLO-D3$_{\uparrow 448}$	86 M	92.9G	448^2	86.3
ELSA-Swin-B	93 M	16.7G	224^2	**84.0**
ELSA-VOLO-D3	87 M	22.3G	224^2	**85.7**
ELSA-VOLO-D3$_{\uparrow 448}$	87 M	98.6G	448^2	**86.6**
CoAtNet-3	168 M	34.7G	224^2	84.5
VOLO-D4	193 M	44.6G	224^2	85.7
VOLO-D4$_{\uparrow 448}$	193 M	194 G	448^2	86.8
Swin-L*	197 M	34.5G	224^2	86.3
Swin-L*$_{\uparrow 384}$	197 M	103.9G	384^2	87.3
SwinV2-L*	197 M	47.5G	256^2	86.9
SwinV2-L*$_{\uparrow 384}$	197 M	115.4G	384^2	87.6
ELSA-Swin-L*	205 M	36.3G	224^2	**86.7**
ELSA-Swin-L*$_{\uparrow 384}$	205 M	106.9G	384^2	**87.6**

Table 8 continued

Model	Params	FLOPs	#Res	Acc@1
CaiT-M36$_{\uparrow 448}$	271 M	248 G	448^2	86.3
CaiT-M48$_{\uparrow 448}$	356 M	330 G	448^2	86.5
VOLO-D5	295 M	72.7G	224^2	86.1
VOLO-D5$_{\uparrow 512}$	295 M	407 G	512^2	87.1
ELSA-VOLO-D5	298 M	78.5G	224^2	**86.3**
ELSA-VOLO-D5$_{\uparrow 512}$	298 M	437 G	512^2	**87.2**

∗Denotes that the model is pretrianed on ImageNet-21K
Simply plugging in ELSA achieves state-of-the-art performance. #Res represents resolutions used in validating / finetuning

tional images, it is very difficult to improve the accuracy of large models under supervised training due to over-fitting. Even so, ELSA still slightly improves VOLO-D5.

It is worth noting that all these records of ELSA-Swin are obtained without modifying any hyperparameters in Swin Transformers, which may not be the optimal setting for our ELSA. For VOLO, we also keep the structure and hyper-parameters of other parts unchanged. Under such a control variable principle, the use of ELSA blocks still achieves state-of-the-art. Redesigning the macro architecture / hyper-parameters manually or by NAS may yield better Pareto performance.

6.2 Object Detection on COCO

Settings. Object detection and instance segmentation experiments are conducted on the COCO dataset. We report the performance on the validation subset, and use the mean average precision (AP) as the metric. We evaluate ELSA-Swin in Mask RCNN / Cascade Mask RCNN (Cai & Vasconcelos, 2018; He et al., 2017), which is a common practice in (Zhang et al., 2021; Yang et al., 2021; Wang et al., 2021b, c; Chen et al., 2021b). Following the common training protocol, we apply multi-scale training, scaling the shorter side of the input from 480 to 800 while keeping the longer side no more than 1333. AdamW (Loshchilov & Hutter, 2017) is adopted as the optimizer with an initial learning rate of 1e-4, weight decay of 5e-2, and batch size of 16. For fair comparisons, all backbones are pretrained using the ImageNet-1K only, and finetuned on the COCO with 1× schedule (12 epochs) and 3× schedule (36 epochs). Our implementation is based on Swin Transformer (Liu et al., 2021) and mmdetection (Chen et al., 2019).

Results. Table 9 lists experimental results. Under the 1× schedule, ELSA-Swin-T and ELSA-Swin-S (noted as ELSA-T / ELSA-S) respectively improve the corresponding baseline by 1.9 AP^b and 1.8 AP^b. They also outperform other methods under a 3× schedule. Mask RCNN with ELSA-Swin-B (noted as ELSA-B) further achieves 48.8 AP^b under 1× schedule and achieves 49.6 AP^b under 3× schedule.

Table 9 Comparison of different backbones on the COCO validation set

Mask RCNN			1× schedule		3× schedule	
Backbone	Params	FLOPs	AP^b	AP^m	AP^b	AP^m
PVT-M	64 M	–	42.0	39.0	44.2	40.5
Focal-T	49 M	291 G	44.8	41.0	47.2	42.7
ViL-S	45 M	218 G	44.9	41.0	47.1	42.7
Swin-T	48 M	267 G	43.7	39.8	46.0	41.6
ELSA-T	49 M	269 G	**45.6**	**41.1**	**47.5**	**42.7**
PVT-L	81 M	–	42.9	39.5	44.5	40.7
Focal-S	71 M	401 G	47.4	42.8	48.8	43.8
ViL-M	60 M	294 G	47.6	43.0	48.9	44.2
Swin-S	69 M	354 G	46.5	42.1	48.5	43.3
ELSA-S	72 M	367 G	**48.3**	**43.0**	**49.2**	**43.6**
ELSA-B	112 M	508 G	**48.8**	**43.1**	**49.6**	**43.9**
Cascade Mask RCNN			1× schedule		3× schedule	
Backbone	Params	FLOPs	AP^b	AP^m	AP^b	AP^m
Swin-T	86 M	745 G	48.1	41.7	50.5	43.7
ELSA-T	86 M	748 G	**49.8**	**43.0**	**51.1**	**44.2**
Swin-S	107 M	838 G	50.3	43.4	51.8	44.7
ELSA-S	110 M	846 G	**51.6**	**44.4**	**52.3**	**45.2**
Swin-B	145 M	982 G	–	–	51.9	45.0
ELSA-B	150 M	987 G	**52.0**	**44.7**	**52.6**	**45.4**

AP^b / AP^m denote the mean average precision of detection / segmentation

Under 1× schedule, Cascade Mask RCNNs with ELSA-Swin-T and ELSA-Swin-S achieve 49.8 AP^b and 51.6 AP^b, which are 1.7 AP^b and 1.3 AP^b higher than their baselines. They also surpass other compared methods under a 3× schedule. Note that, unlike ViL (Zhang et al., 2021) and RegionViT (Chen et al., 2021b), ELSA-Swin does not modify the macro architecture / hyperparameters of Swin Transformer.

6.3 Semantic Segmentation on ADE20K

Settings. We evaluate the semantic segmentation performance of ELSA-Swin on the ADE20K (Zhou et al., 2017), which contains 20K training, 2K validation, and 3K testing images, covering 150 semantic categories. Following (Liu et al., 2021; Han et al., 2022; Yuan et al., 2022), UperNet (Xiao et al., 2018) is selected as the baseline framework. During training, AdamW is adopted as the optimizer. The initial learning rate is set to 6e-5, weight decay is set to 1e-2. All models are trained for 160K iterations with linear learning rate decay, and a linear warmup of 1500 iterations. We use default augmentation settings in mmsegmentation (Contributors, 2020) where the resolution of the input is set to 512 ×

Table 10 Comparison of different backbones on the ADE20K validation set

Backbone	Params	FLOPs	MS mIoU
Swin-T	60 M	945 G	45.8
Focal-T	62 M	998 G	47.0
Twins-SVT-S	54 M	–	47.1
ELSA-Swin-T	61 M	946 G	**47.7**
Swin-S	81 M	1038 G	49.5
Focal-S	85 M	1130 G	50.0
Twins-SVT-B	89 M	–	48.9
ELSA-Swin-S	85 M	1046 G	**50.3**
Swin-B	121 M	1188 G	49.7
Focal-B	126 M	1354 G	50.5
Twins-SVT-L	133 M	–	50.2
ELSA-Swin-B	126 M	1193 G	**50.6**

UperNet (Xiao et al., 2018) is adopted as the framework. All compared backbones are pretrained with the ImageNet-1K only

512. During inference, we perform the multi-scale test. For more details, please refer to (Liu et al., 2021; Contributors, 2020) and our code.

Results. Table 10 shows the mean IoU with multi-scale testing (MS mIoU), model size (Param), and FLOPs of different methods. Results are split into three groups based on the number of model parameters. For fair comparisons, all compared backbones are pretrained using the ImageNet-1K only. UperNet with ELSA-Swin-T is 1.9 higher on MS mIoU than the Swin-T version. Adopting ELSA-Swin-S as the backbone achieves 50.3 MS mIoU, which is 0.8 higher on MS mIoU than Swin-S, and is even better than Swin-B and Twins-SVT-L in the third group. Adopting ELSA-Swin-B as the backbone further achieves 50.6 MS mIoU, which is better than all other compared methods.

7 Conclusions

In this work, we investigate LSA and its counterparts in detail from channel settings and spatial processing to empirically understand the reasons for the unsatisfactory performance of LSA. It is revealed that the relative position embedding and the neighboring filter application are critical reasons why DwConv and dynamic filters perform similarly or better than LSA. Based on these observations, we further propose enhanced local self-attention (ELSA) with Hadamard Attention and the ghost head, which can seamlessly replace LSA and its counterparts in various networks. Experiments show that, without other architecture / hyperparameter modifications, ELSA can consistently improve the baseline, regardless of the model size and tasks, with little overhead being introduced.

Acknowledgements This work was supported by Alibaba Group through the Alibaba Research Intern Program and the National Natural Science Foundation of China (No.61976094).

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., & Agarwal, S. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901.

Cai, Z., & Vasconcelos, N. (2018). Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6154—6162).

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S (2020). End-to-end object detection with transformers. In *ECCV*.

Chen, B., Li, P., Li, B., Li, C., Bai, L., Lin, C., Sun, M., Yan, J., & Ouyang, W. (2021a). Psvit: Better vision transformer via token pooling and attention sharing. arXiv preprint arXiv:2108.03428

Chen, C. F., Panda, R., & Fan, Q. (2021b). Regionvit: Regional-to-local attention for vision transformers. arXiv preprint arXiv:2106.02689

Chen, J., Wang, X., Guo, Z., Zhang, X., & Sun, J. (2021c). Dynamic region-aware convolution. In *CVPR*.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., & Lin, D. (2019) MMDetection: Openmmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155

Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., & Liu, Z. (2020). Dynamic convolution: Attention over convolution kernels. In *CVPR*.

Chen, Y., Dai, X., Chen, D., Liu, M., Dong, X., Yuan, L., & Liu, Z. (2022) Mobile-former: Bridging mobilenet and transformer. In *CVPR*.

Chen, Z., Xie, L., Niu, J., Liu, X., Wei, L., & Tian, Q. (2021d). Visformer: The vision-friendly transformer. In *ICCV*.

Chen, Z., Zhu, Y., Zhao, C., Hu, G., Zeng, W., Wang, J., & Tang, M. (2021e). Dpt: Deformable patch-based transformer for visual recognition. In *ACM MM*, pp 2899–2907

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *PAMI*.

Chu, X., Zhang, B., Tian, Z., Wei, X., & Xia, H. (2021). Do we really need explicit position encodings for vision transformers? arXiv preprint arXiv:2102.10882

Contributors, M. (2020). MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation

Dai, Z., Liu, H., Le, Q.V., & Tan, M. (2021) Coatnet: Marrying convolution and attention for all data sizes. In *NeurIPS*

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *CVPR*

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., & Guo, B. (2021). Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *CVPR*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., & Uszkoreit, J. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*

El-Nouby, A., Touvron, H., Caron, M., Bojanowski, P., Douze, M., Joulin, A., Laptev, I., Neverova, N., Synnaeve, G., & Verbeek, J. (2021) Xcit: Cross-covariance image transformers. In *NeurIPS*.

Fang, Y., Wang, X., Wu, R., & Liu, W. (2021). What makes for hierarchical vision transformer? arXiv preprint arXiv:2107.02174
Gong, C., Wang, D., Li, M., Chandra, V., & Liu, Q. (2021). Improve vision transformers training by suppressing over-smoothing. arXiv preprint arXiv:2104.12753
Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Jégou, H., & Douze, M. (2021). Levit: A vision transformer in convnet's clothing for faster inference. In *ICCV*.
Gu, J., Kwon, H., Wang, D., Ye, W., Li, M., Chen, Y.H., Lai, L., Chandra, V., & Pan, D. Z. (2021). Hrvit: Multi-scale high-resolution vision transformer. arXiv preprint arXiv:2111.01236
Guo, J., Han, K., Wu, H., Xu, C., Tang, Y., Xu, C., & Wang, Y. (2022). Cmt: Convolutional neural networks meet vision transformers. In *CVPR*
Han, Q., Fan, Z., Dai, Q., Sun, L., Cheng, M. M., Liu, J., & Wang, J. (2022). On the connection between local attention and dynamic depth-wise convolution. In *ICLR*
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*
He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *CVPR*.
He, S., Luo, H., Wang, P., Wang, F., Li, H., & Jiang, W. (2021). Transreid: Transformer-based object re-identification. In *ICCV*.
Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., & Lee, Q. V. (2019). Searching for mobilenetv3. In *ICCV*.
Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861
Huang, Z., Ben, Y., Luo, G., Cheng, P., Yu, G., & Fu, B. (2021). Shuffle transformer: Rethinking spatial shuffle for vision transformer. arXiv preprint arXiv:2106.03650
Jia, X., De Brabandere, B., Tuytelaars, T., & Gool, L. V. (2016). Dynamic filter networks. In *NeurIPS*.
Jiang, Z., Hou, Q., Yuan, L., Zhou, D., Jin, X., Wang, A., & Feng, J. (2021). Token labeling: Training a 85.5% top-1 accuracy vision transformer with 56m parameters on imagenet. arXiv preprint arXiv:2104.10858
Li, D., Hu, J., Wang, C., Li, X., She, Q., Zhu, L., Zhang, T., & Chen, Q. (2021a). Involution: Inverting the inherence of convolution for visual recognition. In *CVPR*.
Li, J., Yan, Y., Liao, S., Yang, X., & Shao, L. (2021b). Local-to-global self-attention in vision transformers. arXiv preprint arXiv:2107.04735
Li, S., Chen, X., He, D., & Hsieh, C. J. (2021c). Can vision transformers perform convolution? arXiv preprint arXiv:2111.01353
Lin, M., & Ye, J. (2016). A non-convex one-pass framework for generalized factorization machine and rank-one matrix sensing. In *NeurIPS*.
Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*.
Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*.
Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., & Wei, F. (2022a). Swin transformer v2: Scaling up capacity and resolution. In *CVPR*.
Liu, Z., Mao, H., Wu, C. Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022b). A convnet for the 2020s. In *CVPR*.
Loshchilov, I., & Hutter, F. (2017) Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101
Lu, J., Yao, J., Zhang, J., Zhu, X., Xu, H., Gao, W., Xu, C., Xiang, T., & Zhang, L. (2021). Soft: Softmax-free transformer with linear complexity. In *NeurIPS*.
Ma, N., Zhang, X., Huang, J., & Sun, J. (2020). Weightnet: Revisiting the design space of weight networks. In *ECCV*.
Mehta, S., & Rastegari, M. (2021). Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. arXiv preprint arXiv:2110.02178
Pan, Z., Zhuang, B., He, H., Liu, J., Cai, J. (2022). Less is more: Pay less attention in vision transformers. In *AAAI*.
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & Desmaison, A. (2019). Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.
Polyak, B. T., & Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization, 30*(4), 838–855.
Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., & Dosovitskiy, A. (2021). Do vision transformers see like convolutional neural networks? In *NeurIPS*
Rao, Y., Zhao, W., Tang, Y., Zhou, J., Lim, S. N., & Lu, J. (2022) Hornet: Efficient high-order spatial interactions with recursive gated convolutions. NeurIPS.
Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*
Su, H., Jampani, V., Sun, D., Gallo, O., Learned-Miller, E., & Kautz, J. (2019). Pixel-adaptive convolutional neural networks. In *CVPR*.
Sun, P., Cao, J., Jiang, Y., Zhang, R., Xie, E., Yuan, Z., Wang, C., & Luo, P. (2020). Transtrack: Multiple-object tracking with transformer. arXiv preprint arXiv:2012.15460
Tan, M., Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*.
Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*.
Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. In *ICML*
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *NeurIPS*.
Vaswani, A., Ramachandran, P., Srinivas, A., Parmar, N., Hechtman, B., & Shlens, J. (2021). Scaling local self-attention for parameter efficient visual backbones. In *CVPR*.
Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C. C., & Lin, D. (2019). Carafe: Content-aware reassembly of features. In *ICCV*.
Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C. C , & Lin, D. (2021a). Carafe++: Unified content-aware reassembly of features. PAMI
Wang, P., Wang, X., Luo, H., Zhou, J., Zhou, Z., Wang, F., Li, H., & Jin, R. (2022a). Scaled relu matters for training vision transformers. In *AAAI*.
Wang, P., Wang, X., Wang, F., Lin, M., Chang, S,. Xie, W., Li, H., & Jin, R. (2022b) Kvt: K-nn attention for boosting vision transformers. In *ECCV*.
Wang, W., Xie, E., Li, X., Fan, D. P., Song, K., Liang, D., Lu, T., Luo, P., & Shao, L. (2021b). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*
Wang, W., Yao, L., Chen, L., Cai, D., He, X., & Liu, W. (2021c). Crossformer: A versatile vision transformer based on cross-scale attention. arXiv preprint arXiv:2108.00154
Wightman, R. (2019). Pytorch image models. https://github.com/rwightman/pytorch-image-models, https://doi.org/10.5281/zenodo.4414861
Wu, K., Peng, H., Chen, M., Fu, J., & Chao, H. (2021). Rethinking and improving relative position encoding for vision transformer. In *ICCV*.
Xiao, T., Liu, Y., Zhou, B., Jiang, Y., & Sun, J. (2018). Unified perceptual parsing for scene understanding. In *ECCV*

Xiao, T., Singh, M., Mintun, E., Darrell, T., Dollár, P., & Girshick, R. (2021). Early convolutions help transformers see better. In *NeurIPS*.

Yang, B., Bender, G., Le, Q. V., & Ngiam, J. (2019). Condconv: Conditionally parameterized convolutions for efficient inference. In *NeurIPS*.

Yang, J., Li, C., Zhang, P., Dai, X., Xiao, B., Yuan, L., & Gao, J. (2021). Focal self-attention for local-global interactions in vision transformers. arXiv preprint arXiv:2107.00641

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F. E., Feng, J., & Yan, S. (2021a). Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*

Yuan, L., Hou, Q., Jiang, Z., Feng, J., & Yan, S. (2022). Volo: Vision outlooker for visual recognition. PAMI.

Yuan, Y., Fu, R., Huang, L., Lin, W., Zhang, C., Chen, X., & Wang, J. (2021b). Hrformer: High-resolution transformer for dense prediction. In *NeurIPS*.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J.,& Yoo, Y. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *CVPR*.

Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). Mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412

Zhang, P., Dai, X., Yang, J., Xiao, B., Yuan, L., Zhang, L., & Gao, J. (2021). Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *ICCV*

Zhang, Y., Zhang, J., Wang, Q., & Zhong, Z. (2020). Dynet: Dynamic convolution for accelerating convolutional neural networks. arXiv preprint arXiv:2004.10694

Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., & Zhang, L. (2021). Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., & Torralba, A. (2017) Scene parsing through ade20k dataset. In *CVPR*

Zhou, D., Shi, Y., Kang, B., Yu, W., Jiang, Z., Li, Y., Jin, X., Hou, Q., & Feng, J. (2021a). Refiner: Refining self-attention for vision transformers. arXiv preprint arXiv:2106.03714

Zhou, J., Jampani, V., Pi, Z., Liu, Q., & Yang, M. H. (2021b). Decoupled dynamic filter networks. In *CVPR*.