LcT: Locally-Enhanced Cross-Window Vision Transformer

Canhui Wei
wchswu@163.com
Huiwei Wang
hwwang@swu.edu.cn

College of Electronic and Information Engineering
Southwest University
Chongqing, China

Abstract

Since introducing vision transformers (ViTs), ViT-based models have outperformed convolutional neural networks (CNNs) on various vision tasks. However, competitive ViTs are parameter-heavy and computationally expensive, restricting their applicability to tasks performed on resource-limited devices. In this paper, we propose a lightweight hybrid architecture leveraging the advantage of CNNs and ViTs, called LcT. It achieves the speed-accuracy trade-off with a small number of parameters and computational costs. The proposed LcT is based on two primary operations: 1) It divides the feature map into non-overlapping windows and utilizes the self-attention mechanism to capture cross-window information; 2) It captures local information within each window using a stack of local convolutional layers. The two procedures mentioned above cooperate to complete global information interaction. Our experimental results indicate that LcT is effective and efficient at classification and downstream vision tasks.

1 Introduction

CNNs[15, 19, 21] have reign supreme in computer vision since the 2010s, which can be attributed to their ability to extract image information effectively. For example, CNNs have an inherent spatial inductive bias that matches the natural characteristics of real-world visual tasks. The transformer model[34], which has performed well in Natural Language Processing (NLP), has recently been applied to vision tasks. Without the limitation of the local receptive field in CNNs, ViT-based models learn visual representations on the whole feature map, which brings it the advantage of long-range dependency. Unfortunately, to obtain the same performance as CNNs, ViT-based models have to be scaled up in size, making them parameter-heavy. At the same time, the quadratic relationship with image size makes it unsuitable for downstream tasks, such as object detection and semantic segmentation.

Some ViT variants have been proposed to reduce the complexity, for example, constraining the attention region (e.g., Swin[24], Shuffle transformer[20], D-DW-Conv[13], CSWin[8] and Focal[41]), reducing transformer dimension (e.g., ResT[46], Twins[4] and CMT[12]), and reducing the number of queries, keys, or values (e.g., CvT[38], PVT[35] and PVTv2[36]). However, they are still far from being deployed to resource-limited devices. In addition, compared to CNNs, more complex training strategies (e.g., AdamW[25] optimizer and stochastic depth[18]), large-scale datasets (e.g., ImageNet 22k and JFT-300M), and strong data augmentation techniques (e.g., Rand Augment[6], multi-scale training[26], Mixup[45] and Cutmix[44]) are also the basis for optimizing ViT-based models. Heavy data augmentation may harm the model's generalization capacity when transferring models pre-trained on large datasets to other downstream datasets[47].

In some real-world visual tasks that require low latency but run on devices with limited computational capability, CNNs still dominate their backyard. It is crucial to create a lightweight and general-purpose model for these applications.

In this paper, We present LcT, a hybrid structure better suited to complicated vision applications. It takes advantage of ViT's ability to model cross-window information and CNN's ability to learn local features. Specifically, a feature map is evenly partitioned into non-overlapping windows. The unfolding operation packs patches at the corresponding location in each window into M^2 groups, as shown in Figure 1(c). The patches in each group are computed by self-attention to model cross-window information. The folding operation will restore patches to the full feature map based on their original positions. A local 3×3 standard convolutional layer and inverted residual blocks (IRB)[29] are used to model the local information within each window.

The proposed LcT demonstrates competitive performance on various vision tasks, such as image classification, object detection, and semantic segmentation. For instance, it achieves a top-1 accuracy of 79.4% on ImageNet-1K classification at 6.5 million parameters, surpassing MobileViT[26] and CoaT-Lite[40] by a clear margin. We note that the superiority of LcT is not limited to theoretical metrics such as parameters or FLOPs. The LcT continues to outperform state-of-the-art models in terms of throughput on GPU. The benefit of LcT's general-purpose architecture is further highlighted when transferring from image classification to dense tasks. It gains 3.7 AP when MobileViT is replaced with LcT as the backbone in SSDLite[23]. On PASCAL VOC 2012 dataset, LcT surpasses MobileViT by 1.9 mIoU when substituting the backbone in DeepLabV3[2].

2 Related Work

2.1 Combination of CNN and ViT

Vanilla ViT[9] directly cuts an original image into non-overlapping patches and applies the multi-headed self-attention (MSA) mechanism to model information between each patch.

In contrast to the original ViT, which only uses linear layers, the hybrid architecture integrates CNNs in different locations or replaces crucial components to achieve better performance and robustness. CeiT[42] uses a convolutional layer and a max-pooling layer as a substitute for patchify stem to capture low-level information in original images and insert a depth-wise separable convolution[3] into the feed-forward network (FFN) to model local information. ViT$_C$[39] replaces the straightforward tokenization with a convolutional stem and makes optimizing it easier. CvT[38] introduces convolutional projection to replace position-wise linear projection for the attention operation. CoaT[40] applies a convolutional position encoding as an alternative to absolute position embedding in ViT[9] and DeiT[33].

2.2 Effort to lower latency

As illustrated in Figure 1(a), all patches from the same feature map will participate in the G-MSA in ViT[9]. The overall computational complexity of the G-MSA module is:

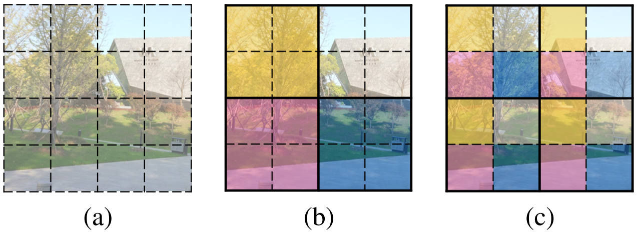


Figure 1: An illustration of three self-attention modes. **Patch**: outlined in dotted lines; **Window**: outlined in solid lines. (a) Global MSA (G-MSA): all patches of a feature map perform the self-attention operation, which is used in ViT[9]. (b) Local-Window MSA (W-MSA): the self-attention is constrained to each window, which is proposed by Swin[24]. (c) Cross-Window MSA (CW-MSA): the patches (with the same color) from different windows with the same corresponding position in the window perform the self-attention operation, which is proposed by MobileViT[26], and we will adopt this method.

$$\mathcal{O}(\text{G-MSA}) = 4hwC^2 + 2(hw)^2C \tag{1}$$

Here, the feature map contains $h \times w$ patches, and C is the attention dimension. Its computational complexity is quadratic to feature size, which makes it unsuitable for processing high-resolution images or hierarchical representations. To overcome this problem, Swin[24] introduces the local-window self-attention mechanism to restrict the attention region, as shown in Figure 1(b). The overall computational complexity of a W-MSA module is:

$$\mathcal{O}(\text{W-MSA}) = 4hwC^2 + 2M^2hwC \tag{2}$$

Here, $M \times M$ is the window size. The self-attention is constrained to each nonoverlapping window to reduce computational complexity. When the window size is fixed, the complexity becomes linear to the image size. However, a great number of transformer blocks are essential to exchange information between adjacent windows and shifted strategy used to build window connections is inconvenient to realize. Similarly, CSWin[8] computes self-attention in the cross-shaped region.

Although these efforts have boosted efficiency, they are still far from being implemented for tasks with limited resources. MobileViT[26] applies CW-MSA to exchange cross-window information and convolutions to model local-window information, as shown in Figure 1(c). The overall computational complexity of the CW-MSA module is:

$$\mathcal{O}(\text{CW-MSA}) = 4hwC^2 + 2(hw/M^2)hwC \tag{3}$$

Despite its outstanding classification results, MobileViT's thin and shallow design provides a potential for improvement on downstream tasks. There are at least three problems with it; 1) The shallow and narrow architecture. To achieve adequate multi-scale features for detectors like SSDLite[23], three-stage MobileViT needs to construct more extra layers. On the other hand, these extra layers have poorer modeling power than the main stages; 2) The small size of the window. Small kernel convolution cannot sufficiently aggregate local information and results in performance deterioration if the kernel is smaller than the window size. According to the results in MobileViT, substituting the ideal window size of [2, 2, 2]

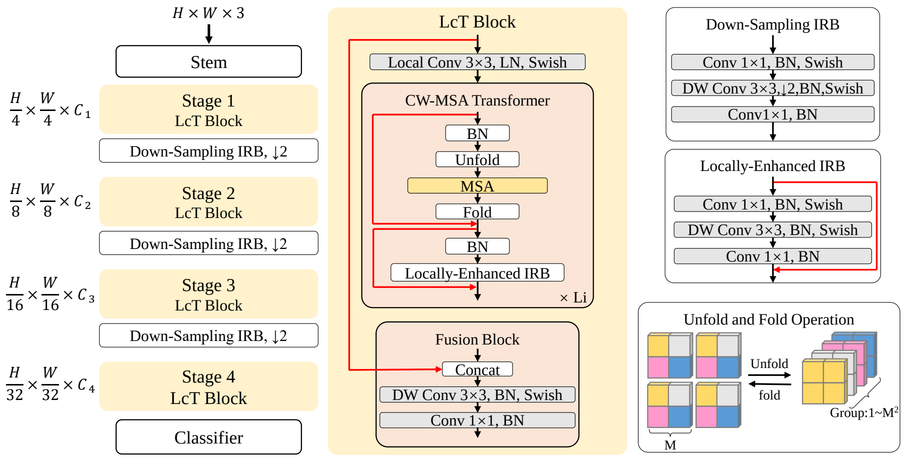


Figure 2: The overall hierarchical architecture of LcT and its sub-block. The unfold operation extracts patches from different windows at the corresponding locations. Then, self-attention operations are performed individually for each group of patches. The fold operation restores all patches to the complete feature map according to their original positions.

with [4, 4, 4] or [8, 4, 2] would induce a slight decline in ImageNet-1K accuracy (78.4% to 77.6% and 77.3% respectively). This indicates that the window size cannot exceed 3×3. However, computing complexity is inversely proportional to the window size, as shown in Formula 3. 3) The high rate of downsampling before the main stages. MobileViT uses an eight-fold downsampling rate to solve the above problem to reduce the input image's resolution. This operation results in the loss of precise information about small objects during detection and segmentation.

Naturally, it remains an open question: *Is it possible to design efficient and effective networks capable of both classification and downstream tasks?*

Combining the strengths of CW-MSA and CNNs is a sensible way to design modern network architecture. Our work will show that hybrid LcT can thoroughly utilize the potency of the convolutional layer to allow us a large window size while restricting the burden rendered by large kernels.

3 Method

3.1 Overall Architecture

The overall architecture of LcT presents in Figure 2. For the $H \times W \times 3$ input image, LcT obtains $\frac{H}{4} \times \frac{W}{4}$ feature maps through the stem structure, which includes four 3×3 successive convolutional layers. In four main stages, each LcT block consists of a local 3×3 standard convolutional layer to learn local information, as MobileViT[26] does. We utilize locally-enhanced inverted residual blocks (Locally-Enhanced IRB)[29] instead of MLPs to learn local representations further and gradually enlarge the convolution receptive field from 3×3 to a larger size in the CW-MSA transformer block. A depth-wise (DW) and point-wise (PW) convolution block are used to fuse the concatenated features after these transformer blocks.

With an exception to the last stage, a down-sampling IRB is applied to reduce the resolution of the features and increase the channel dimension after the LcT block. Four stages will produce multi-scale feature maps with resolutions of $\frac{H}{4} \times \frac{W}{4}$, $\frac{H}{8} \times \frac{W}{8}$, $\frac{H}{16} \times \frac{W}{16}$ and $\frac{H}{32} \times \frac{W}{32}$ respectively, hence our model can be easily served as a backbone for task-specific models.

We note that LayerNorm (LN) is replaced by BatchNorm (BN) with an exception to the local 3×3 standard convolutional layer before the CW-MSA transformer blocks, which is quite different from most transformer designs.

3.2 Cross-Window Transformer

Cross-Window Multi-Headed Self-Attention (CW-MSA) The G-MSA used in vanilla ViT[9], W-MSA employed in Swin[24], and CW-MSA adopted in MobileViT[26] all suffer from their dilemmas. To alleviate these problems, we will use locally-enhanced CW-MSA with a large window size to improve efficiency while suppressing performance degradation.

Considering the embedded feature $X_e \in R^{(h \times w) \times C}$, here $N = h \times w$ is the number of patches. In the CW-MSA transformer, the feature is evenly partitioned into nonoverlapping windows by unfold operation so that N patches are divided into M^2 groups. Here, M is the window size. Each group contains $L = hw/M^2$ patches coming from the corresponding location in each window, as illustrated in Figure 1(c). Then the MSA will perform within each group and linearly projects it to several heads with dimension d, respectively. The process is:

$$\begin{aligned} & X \xrightarrow{\text{unfold}} [X^1, X^2, ..., X^{M^2}], \text{where} X^k \in R^{(hw/M^2) \times C} \\ & Y^k = \text{MSA}(X^k), \text{where} k = [1, 2, ..., M^2] \\ & [Y^1, Y^2, ..., Y^{M^2}] \xrightarrow{\text{fold}} Y \end{aligned} \tag{4}$$

Where $h \times w$ is the feature height and width for patch level, $M \times M$ is the window size, and C is the channel dimension.

Considering the large resolution of the feature map in the early layers, we use a correspondingly large window in the first stage and gradually reduce the window size in the following stages. This design is quite different from the previous work. In four stages, the window size is set to [7, 4, 2, 1], [8, 4, 2, 1], and [8, 4, 2, 2] for classification object detection and semantic segmentation task, respectively. We note that the window size does not change the structure of the model so that we can use the pre-trained backbone to finetune the downstream model, even if we use different window-size values in different tasks.

Locally-Enhanced IRB Since we use a 7×7 or 8×8 window in the first stage and the local convolution is only 3×3, we must find a mechanism to enhance local representation. Instead of using linear layers to aggregate MSA's output, we replace MLP with an inverted residual block, as illustrated in Figure 2. Rather than using a 5×5 or 7×7 kernel size, which is expensive in terms of computation or parameter, previous work[31] has proved that two 3×3 convolutional layers can replace a 5×5 convolutional layer to obtain the same receptive field. Specifically, the first stage of all variants contains two CW-MSA transformer blocks and indirectly stacks three 3×3 convolutions (e.g., two locally-enhanced IRBs and one local standard convolutional layer). Since we have stacked three 3×3 convolutional layers, a 7×7 local receptive field has been created to cover the first stage's 7×7 window. Similarly, the same operations are applied to other stages. Although it is still slightly smaller than the 8×8 window on downstream tasks, the slight performance degradation is acceptable.

Model	#Channels	#Blocks	#Heads	#Param.	Flops
LcT-Tiny	Stem: [16,24,36,36] Stages: [36,72,108,144]	[2,2,4,2]	[3,6,9,12]	1.89M	0.63G
LcT-Small	Stem: [16,32,48,48] Stages: [48,96,144,192]	[2,2,4,2]	[2,4,6,8]	3.22M	1.09G
LcT-Base	Stem: [16,32,64,96] Stages: [96,144,192,240]	[2,2,6,2]	[4,8,6,10]	6.53M	2.65G

Table 1: Details of the LcT variants. #Blocks: number of CW-MSA transformer blocks in each stage.

Task	Classification	Object detection	Semantic segmentation (pretraining/finetuning)
Dataset	ImageNet-1K	MS-COCO	MS-COCO/VOC 2012 AUG
Epoch or iteration	300 epochs	60 epochs	50 epochs/40k iterations
Batch size	700	32	16/16
Base LR	0.000684	0.0002	6e-5/1e-5
Optimizer	AdamW	AdamW	AdamW/AdamW
Weight decay	0.05	0.03	0.02/0.01
Data augmentation	Random crop, resize, horizontal flipping	MinIoURandomCrop, resize, random flipping, photometric distortion	Random crop, resize, random flipping, photometric distortion
Scheduler	Cosine	Step	Cosine/Cosine
Decay epoch	None	[35, 45, 52, 57]	None/None
Warm-up scheduler	Linear	Linear	Linear/Linear
Warm-up iteration	20 epochs	600 iterations	1500 iterations/1500 iterations
Stochastic depth	0.1	0.1	0.1/0.1

Table 2: Training settings for image classification, object detection, and semantic segmentation .

The above work perfectly solves the issue: a small convolution kernel can well aggregate local features, and a low down-sampling rate can retain richer spatial information on dense prediction tasks.

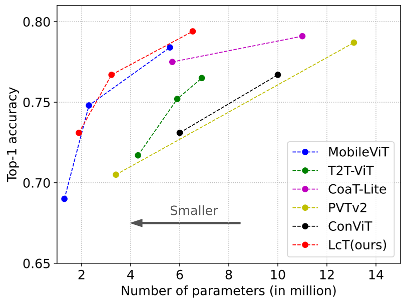
Comparisons to existing designs Some previous works[12, 42] have employed the IRB to replace the original MLP in the feed-forward neural network. However, their goals are fundamentally different from those of our proposed LcT. Our model uses CW-MSA to exchange cross-window information, so the purpose of the IRB is clearer and more specific (i.e., aggregating information within a window and expanding the receptive field).

3.3 Architecture Details and Variants

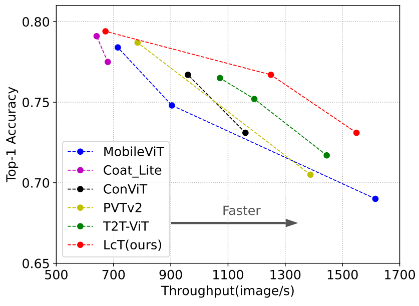
For a fair comparison with other models, we constructed three variants, named LcT-Base, LcT-Small, and LcT-Tiny, as shown in Table 1. The expansion ratio is 2 in the locally-enhanced IRB for all variants. The expansion ratio is set to 4, 3, and 2 for LcT-Base, LcT-Small, and LcT-Tiny in the down-sampling IRB. The classifier contains a 1×1 convolution that raises the dimensionality (The ratio is 4, 3, and 3 for LcT-Base, LcT-Small, and LcT-Tiny, respectively), a global average pooling layer, and a fully connected layer that maps the output to the category vector.

Model	Input	#param.	FLOPs	#Aug.	Throughput (images/s)	Top-1
MobileViT-XXS[26]	256^2	1.3M	0.4G	Basic	1615	69.0
PVTv2-B0[36]	224^2	3.4M	0.6G	Strong	1388	70.5
T2T-ViT-7[43]	224^2	4.3M	1.1G	Strong	1445	71.7
LcT-Tiny(ours)	224^2	1.9M	0.6G	Basic	1549	**73.1**
MobileViT-XS[26]	256^2	2.3M	1.1G	Basic	904	74.8
ConViT-Tiny[10]	224^2	6.0M	1.0G	Strong	1161	73.1
ConViT-Tiny+[10]	224^2	10.0M	2.0G	Strong	960	**76.7**
T2T-ViT-10[43]	224^2	5.9M	1.5G	Strong	1192	75.2
T2T-ViT-12[43]	224^2	6.9M	1.8G	Strong	1072	76.5
LcT-Small(ours)	224^2	3.2M	1.1G	Basic	1250	**76.7**
MobileViT-S[26]	256^2	5.6M	2.0G	Basic	715	78.4
PVTv2-B1[36]	224^2	13.1M	2.1G	Strong	784	78.7
CoaT-Lite Tiny[40]	224^2	5.7M	1.6G	Strong	680	77.5
CoaT-Lite Mini[40]	224^2	11.0M	2.0G	Strong	641	79.1
LcT-Base(ours)	224^2	6.5M	2.7G	Basic	672	**79.4**

Table 3: Comparison with ViTs on ImageNet-1K validation set. Throughput is measured on an RTX8000 GPU. The results indicate that LcT is less dependent on data augmentation and may obtain better transfer ability according to previous work[47].



(a)



(b)

Figure 3: Comparison of the number of parameters and throughput on ImageNet-1K classification. More details are in Table 3.

4 Experiments

We benchmark our proposed LcT on image classification, object detection, and semantic segmentation. First, we evaluate LcT's performance on the classification task and compare it with the current state-of-the-art models. We also report GPU throughput (images/second) to show real-world efficiency. Then, our experimental results demonstrate that LcT can be conveniently integrated into various lightweight frameworks and outperform other models on downstream tasks.

4.1 Image Classification on ImageNet-1K

Implementation details: For image classification, we evaluate our model on the ImageNet-1K[7] benchmark, which contains 1.28M images for training and 50K images for validation from 1000 classes. For training, We mostly follow [24] to train LcTs on 224×224 resolution, and more details are in Table 2. We evaluate the models on the ImageNet-1K validation set with single crop top-1 accuracy. Models with Exponential Moving Average[27] (decay rate = 0.9999) are used to test.

Comparison with state-of-the-art methods: Table 3 shows the results of LcT and various ViTs and is divided into three parts based on throughput. Results are based on image resolution of 224×224, except for MobileViTs with an input of 256×256 according to their own implementation. All of them are trained without distillation methods and extra data. Here, basic data augmentation denotes random resized cropping and horizontal flipping, while strong data augmentation denotes Mixup[45], CutMixup[44], RandAugment[6], and basic data augmentation. Obviously, LcTs are better and more robust than other variants. LcT-Base achieves better top-1 accuracy than MobileViT[26]/ PVTv2-B1[36]/ CoaT-Lite Mini[40] (top-1 accuracy: 79.4% vs. 78.4%/ 78.7%/ 79.1%, parameters: 6.5M vs. 5.6M/ 13.1M/ 11.0M). The accuracy-parameter curves in Figure 3(a) demonstrate the superiority of LcT.

Inference speed on GPU: Throughput (images/second on a single GPU) is measured on an RTX8000 GPU with a batch size of 128 following [33] and [24]. For other models, we use their original implementations, except for MobileViT and ConViT from timm[37] because they do not provide a convenient interface. In some cases, FLOPs and the number of parameters as an indirect metric do not accurately reflect a model's actual speed. For example, in Table 3, LcT-Base has more FLOPs than both MobileViT-S and CoaT-Lite Mini, but they still achieve a similar throughput. This is because we use fewer LayerNorms. For example, we only use one LayerNorm in each stage. In contrast to LayerNorm, BatchNorm can be combined with convolution for acceleration, resulting in lower latency. Based on the results of our ablation experiments, our trade-off between LayerNorm and BatchNorm significantly improves the throughput while maintaining performance. Figure 3(b) plots the speed-accuracy curve, and our model consistently outperforms the state-of-the-art models.

4.2 Object Detection on COCO

Implementation details: We evaluate LcTs on MS-COCO 2017 dataset[22], which contains 118K training, 5K validation, and 20K test-dev images. Specifically, we applied LcTs as the backbone in SSDLite[29] detector. We point out that this task sets the window size to [8, 4, 2, 1] for each stage, respectively. During training and evaluation, the image resolution is set to 320×320. Our experiments are conducted in mmdetection[1], and more details are in Table 2.

Results: Table 4 shows the results of metric of mAP@0.5:0.05:0.95 measured on the COCO validation 2017 (mini-val) and test-dev 2017 (test-dev). Depending on our effective structural design discussed in Section 3, we make significant improvements in both comparing CNN backbones and ViT backbone. Specifically, SSDLite-LcT-Base gains +5.9 and +6.2 mAP over SSD300-VGG[30] and SSD300-ResNet-50[15] but has fewer FLOPs and fewer parameters. In addition, with a few more parameters, LcT-Base and LcT-Small surpass the previous light-weight MobileViT-S[26] and MobileViT-XS by +3.7 and +2.1 mAP as backbones in SSDLite.

The results have shown that the number of main stages and appropriate stride are the key factors affecting the performances on dense prediction tasks. As a result, the structure design of LcTs is more suitable as a general-purpose backbone.

Method	Backbone	Input	Param.	FLOPs	test-dev	mini-val
SSDLite	MobileNetV1[17]	320^2	5.1M	1.3G	22.2	-
	MobileNetV2[29]	320^2	4.3M	0.8G	22.1	†21.3
	MobileNetV3[16]	320^2	4.9M	0.6G	22.0	-
	MnasNet-A1[32]	320^2	4.9M	0.8G	23.0	-
	MobileViT-XS[26]	320^2	2.7M	-	-	24.8
	LcT-Small(ours)	320^2	3.5M	2.7G	-	**26.9**
	MobileViT-S[26]	320^2	5.7M	-	-	27.7
	LcT-Base(ours)	320^2	6.7M	6.1G	-	**31.4**
SSD	ResNet-50[15]	300^2	22.9M	-	-	‡25.2
	VGG[30]	300^2	34.3M	34.4G	-	†25.5
	VGG[30]	512^2	36.0M	98.8G	-	†29.5

Table 4: Detection results on COCO dataset. † means this result is form mmdetecton[1] and ‡ means this result is form [26].

Backbone	Input	Param.	FLOPs	mIOU
MobileViT-XS[26]	512^2	2.9M	-	77.1
MobileNetV1[17]	512^2	11.2M	14.2G	75.3
MobileNetV2[29]	512^2	4.5M	5.8G	75.7
MobileViT-S[26]	512^2	6.4M	-	79.1
LcT-Small(ours)	512^2	3.8M	9.1G	**79.4**
ResNet101[15]	512^2	58.2M	81.0G	80.5
LcT-Base(ours)	512^2	7.4M	19.5G	**81.0**

Table 5: Semantic segmentation results w/ DeepLabV3 on VOC2012 validation set.

4.3 Semantic Segmentation on VOC

Implementation details: We experiment on the VOC 2012 dataset[11] covering 21 semantic categories. We adopt DeepLabV3[2] based on the implementation from mmsegmentation[5] as our base framework. Following [2], extra annotations[14] and dataset[22] are used. First, following the official PyTorch repository[28], We select objects from the COCO 2017 dataset with the same category as the VOC 2012 dataset and pre-train DeepLabV3-LcTs. Then we fine-tune models on the VOC 2012 AUG dataset[14] and evaluate them on the VOC 2012 validation set. The input image is scaled to 512×512 on training and evaluation. More Details are in Table 2. In evaluating, the metric of mIOU is measured on a single scale.

Result: As shown in Table 5, Our LcT consistently outperforms other models when used as the backbones on DeepLabV3. Specifically, LcT-Base/LcT-Small brings 2.3/1.9 mIOU improvements over MobileViT-S[26]/MobileViT-XS[26], respectively. Compared to the heavy-weight backbone, we obtain a slight improvement (+0.5 mIOU) but save a lot of parameters and computational consumption (LcT-Base vs. ResNet101[15]: 7.4M vs. 58.2M: 19.5G vs. 81.0G).

4.4 Ablation Experiments

In this section, we conduct ablation experiments to explore the impact of critical components. Considering the resource constraints, we train LcT-Small for 50 epochs and then evaluate its performance on ImageNet-1k. We reduce the learning rate and warm-up epoch to 0.000273 and 10, respectively, and keep other hyper-parameters.

Window Sizes. In Table 6, part 1 shows the comparison results for different window sizes. When using smaller windows ([7, 7, 7, 1] $\rightarrow$ [7, 4, 2, 1] for four stages, respectively), there was a significant gain (+0.8%) in accuracy, which means that stacking convolution

Method		#Param.	FLOPs	Throughput (images/s)	Top-1
Windows size	[7, 7, 7, 1]	3.2M	1.07G	1263	63.74
	[7, 4, 2, 1]	3.2M	1.09G	1250	64.54(+0.8)
	[4, 2, 2, 1]	3.2M	1.21G	1046	64.56(+0.82)
FFN type	MLP	3.2M	1.08G	1415	64.09
	IRB	3.2M	1.09G	1250	64.54(+0.45)
Normalization	All BN	3.2M	1.09G	1297	63.84
	All LN	3.2M	1.09G	1077	64.46(+0.62)
	LN+BN	3.2M	1.09G	1250	64.54(+0.7)

Table 6: Ablation experiments exploring the role of window size, ffn type, and normalization.

increases the receptive field is an indirect and compromised approach. However, when we reduce the window size to [4, 2, 2, 1], the performance does not improve further, which means that the locally-enhanced module enables us to adopt a larger window and thus increase the speed ([7, 4, 2, 1] vs. [4, 2, 2, 1]: 1250 vs. 1046 image/s).

FFN type. In Table 6, part 2 compares the locally-enhanced IRB and the MLP in the feedforward neural network. We observe a +0.45% boost using locally-enhanced IRB instead of MLP, demonstrating that the local enhancement module plays an important role.

Normalization. The LN is placed before MSA and FFN in the original ViT and most of its variants to stabilize the tokens distribution by normalizing the channel dimension. BN is the standard normalization method for convolution to stabilize the distribution by normalizing the batch dimension. As shown in Table 6, replacing all LNs with BNs leads to a significant accuracy degradation, whereas replacing all BNs with LNs results in a speed loss. Considering the balance of accuracy and speed, we insert LN into the local 3×3 convolution and utilize BN before MSA and locally-enhanced IRB. The results show that this strategy achieves a balance of accuracy and speed.

Other ways to improve the receptive field. Atrous convolution is another method that does not lose image information while increasing the perceptual field. However, we observe severe performance degradation when using atrous convolution in FFN. For example, using atrous convolution with rate=3 in IRB results in a 0.88% drop compared to the baseline (64.54% $\rightarrow$ 63.76%).

5 Conclusion

In this work, we propose a lightweight, general-purpose vision architecture that overcomes the shortcomings of some previous work by fusing the cross-window self-attention mechanism and CNN. Our extensive experiments show that LcT not only outperforms existing state-of-the-art models for classification tasks but also dramatically improves the performance of downstream tasks such as object detection and semantic segmentation. As a lightweight ViT, LcT has few parameters and computational effort, filling some gaps in the ViT family for applications on resource-limited devices.

Acknowledgment

This work was supported by Chongqing Natural Science Foundation Projects (cstc2020jcyj-msxmX0057).

References

[1] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[4] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[5] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

[6] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] 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. *arXiv preprint arXiv:2107.00652*, 2021.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *In International Conference on Learning Representations*, 2021.

[10] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021.

[11] Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[12] Jianyuan Guo, Kai Han, Han Wu, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Cmt: Convolutional neural networks meet vision transformers. *arXiv preprint arXiv:2107.06263*, 2021.

[13] Qi Han, Zejia Fan, Qi Dai, Lei Sun, Ming-Ming Cheng, Jiaying Liu, and Jingdong Wang. Demystifying local vision transformer: Sparse connectivity, weight sharing, and dynamic weight. *arXiv preprint arXiv:2106.04263*, 2021.

[14] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 international conference on computer vision*, pages 991–998. IEEE, 2011.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[18] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[20] Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu. Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arXiv preprint arXiv:2106.03650*, 2021.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[24] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[25] I Loshchilov and F Hutter. Decoupled weight decay regularization, 7th international conference on learning representations, iclr. *New Orleans, LA, USA, May*, (6-9):2019, 2019.

[26] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021.

[27] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[28] PyTorch. Torchvision semantic segmentation. `https://github.com/pytorch/vision/tree/main/references/segmentation`, 2022.

[29] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[32] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[33] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[35] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021.

[36] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, pages 1–10, 2022.

[37] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[38] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22–31, 2021.

[39] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *Advances in Neural Information Processing Systems*, 34:30392–30400, 2021.

[40] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9981–9990, 2021.

[41] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*, 2021.

[42] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 579–588, 2021.

[43] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 558–567, 2021.

[44] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[45] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[46] Qinglong Zhang and Yu-Bin Yang. Rest: An efficient transformer for visual recognition. *Advances in Neural Information Processing Systems*, 34, 2021.

[47] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.