Less is More: Pay Less Attention in Vision Transformers

Zizheng Pan, Bohan Zhuang*, Haoyu He, Jing Liu, Jianfei Cai

Data Science & AI, Monash University, Australia
{zizheng.pan, bohan.zhuang, haoyu.he, jing.liu1, jianfei.cai}@monash.edu

Abstract

Transformers have become one of the dominant architectures in deep learning, particularly as a powerful alternative to convolutional neural networks (CNNs) in computer vision. However, Transformer training and inference in previous works can be prohibitively expensive due to the quadratic complexity of self-attention over a long sequence of representations, especially for high-resolution dense prediction tasks. To this end, we present a novel Less attention vIsion Transformer (LIT), building upon the fact that the early self-attention layers in Transformers still focus on local patterns and bring minor benefits in recent hierarchical vision Transformers. Specifically, we propose a hierarchical Transformer where we use pure multi-layer perceptrons (MLPs) to encode rich local patterns in the early stages while applying self-attention modules to capture longer dependencies in deeper layers. Moreover, we further propose a learned deformable token merging module to adaptively fuse informative patches in a non-uniform manner. The proposed LIT achieves promising performance on image recognition tasks, including image classification, object detection and instance segmentation, serving as a strong backbone for many vision tasks. Code is available at https://github.com/zip-group/LIT.

Introduction

Transformers have made substantial strides in natural language processing (NLP) (*e.g.*, (Vaswani et al. 2017; Devlin et al. 2019)) and recently in the computer vision (CV) field (*e.g.*, (Dosovitskiy et al. 2021; Touvron et al. 2021a)). Inspired by the pyramid design (Lin et al. 2017a) in CNNs, recent hierarchical vision Transformers (HVTs) (Wang et al. 2021; Liu et al. 2021; Wu et al. 2021; Yan et al. 2021) divide transformer blocks into several stages and progressively shrink feature maps as the network goes deeper. However, high-resolution feature maps in the early stages result in long token sequences, which brings huge computational cost and memory consumption due to the quadratic complexity of self-attention. For instance, a feature map of size $56 \times 56 \times 96$ costs 2.0G FLOPs in one Multi-head Self-Attention (MSA) (Vaswani et al. 2017), while the entire model of ResNet-18 (He et al. 2016) only requires 1.8G FLOPs. Such a huge computing cost makes it difficult to apply Transformers into broad computer vision tasks.

Several emerging efforts have been made to reduce the computational cost in the early stages of HVTs. For example, some works (Wu et al. 2021; Vaswani et al. 2021) reduce the number of self-attention heads in an MSA layer or further decreasing the number of Transformer blocks (Dosovitskiy et al. 2021). Another line of works proposes to trade-off accuracy and efficiency for MSA via heuristic approximation, such as spatial reduction attention (SRA) (Wang et al. 2021) and shifted window based multi-head self-attention (SW-MSA) (Liu et al. 2021). There are also studies simply employ convolutional layers (Srinivas et al. 2021; Graham et al. 2021) when the resolution of feature maps are considerably large. However, how much the early adoption of self-attention layers really contributes to the final performance remains unclear.

In this paper, we present a **L**ess attention v**I**sion **T**ransformer (**LIT**) to address the aforementioned problem for HVTs. Specifically, we propose to exclusively use MLP layers to capture local patterns in the early stages while introducing MSA layers with sufficient number of heads to handle long range dependencies in the later stages. The motivation comes from two aspects. First, previous studies in both CNNs and Transformers have shown that shallow layers focus on local patterns and deeper layers tend to capture high-level semantics or global relationships (Wu, Su, and Huang 2019; Hou et al. 2019; Tenney, Das, and Pavlick 2019), arising the question of whether using self-attention at the early stages is necessary. Second, from the theoretical perspective, a self-attention layer with sufficient heads applied to images can express any convolutional layer (Cordonnier, Loukas, and Jaggi 2020). However, fewer heads in an MSA layer theoretically hinder the ability of approximating a convolutional layer with a large kernel size, where the extreme case is as expressive as a 1×1 convolution that can be viewed as a standard FC layer applied to each pixel independently. While recent HVTs adopt very few heads at the early stages to deliver pyramid representations, we argue that this is not optimal as such setting introduces high computational and memory cost but brings minor benefits.

To be emphasized, by exploiting MLP blocks in the early stages, the model avoids the huge computational cost and memory footprint arising from self-attention on high-

*Corresponding author.

resolution feature maps. Moreover, applying self-attention in the later stages to capture long range dependencies is quite efficient due to the progressive shrinking pyramid. Our comprehensive results show that such a simple architecture design brings a sweet spot between model performance and efficiency.

Furthermore, recent HVTs either adopt a standard convolutional layer or a linear projection layer to merge nearby tokens (Wang et al. 2021; Liu et al. 2021), aiming to control the scale of feature maps. However, such methods hinder the representational power for vision Transformers to model geometric transformations, considering that not every pixel equally contributes to an output unit (Luo et al. 2016). To this end, we propose a Deformable Token Merging (DTM) module, inspired by deformable convolutions (Dai et al. 2017; Zhu et al. 2019), where we learn a grid of offsets to adaptively augment the spatial sampling locations for merging neighboring patches from a sub-window in a feature map. In this way, we can obtain more informative downsampled tokens for subsequent processing.

Our contributions can be summarized as follows. First, we identify the minor contribution of the early MSA layers in recent HVTs and propose a simple HVT structure with pure MLP blocks in the early stages. Second, we propose a deformable token merging module to adaptively merge more informative patches to deliver hierarchical representations, with enhanced transformation modeling capability. Finally, we conduct extensive experiments to show that the proposed LIT performs favorably against several state-of-the-art vision Transformers with similar or even reduced computational complexity and memory consumption.

Related Work

Vision Transformers. Vision Transformers are models which adopt the self-attention mechanism (Vaswani et al. 2017) into CV tasks. Recent works towards Vision Transformers either follow a hybrid architecture that combines convolution and self-attention (Carion et al. 2020; Srinivas et al. 2021), or design a pure self-attention architecture without convolution (Parmar et al. 2019; Hu et al. 2019). More recently, Dosovitskiy *et al.* (Dosovitskiy et al. 2021) propose a Vision Transformer (ViT) which achieves promising results on ImageNet. Since then, a few subsequent works have been proposed to improve ViT from different aspects. For example, some works (Yuan et al. 2021b; Li et al. 2021) seek to bring locality into ViT as they find ViT failed to model the important local structures (*e.g.*, edges, lines). Another line of works aims to explore deeper architectures (Zhou et al. 2021; Touvron et al. 2021b) by stacking more Transformer blocks. Some studies (Chen et al. 2021a,b) also try to search a well-performed ViT with neural architecture search (NAS).

There is also a prevailing trend to introduce hierarchical representations into ViT (Pan et al. 2021; Yuan et al. 2021a; Wang et al. 2021; Liu et al. 2021; Wu et al. 2021; Heo et al. 2021; Vaswani et al. 2021). To do so, these works divide Transformer blocks into several stages and downsample feature maps as the network goes deeper. However, high-resolution feature maps in the early stages inevitably result in high computational and memory costs due to the quadratic complexity of the self-attention module. Targeting at this problem, Wang *et al.* (Wang et al. 2021) propose to reduce the spatial dimensions of attention's key and value matrices. Liu *et al.* (Liu et al. 2021) propose to limit self-attention in non-overlapped local windows. However, these replacements seek to shrink the global attention maps for efficiency. In this paper, we elaborately design the shallow layers with pure MLPs, that are powerful enough to encode local patterns. This neat architecture design keeps the capability for modelling global dependencies in the later stages while easing the prohibitively expensive complexity introduced by high-resolution feature maps, especially in the dense prediction tasks.

Deformable Convolutions. Deformable convolutions (DC) are initially proposed by Dai *et al.* (Dai et al. 2017) in object detection and semantic segmentation tasks. Different from the regular sampling grid of a standard convolution, DC adaptively augments the spatial sampling locations with learned offsets. One following work by Zhu *et al.* (Zhu et al. 2019) improves DC with a modulation mechanism, which modulates the input feature amplitudes from different spatial locations. With the advantage on modeling geometric transformations, many works adopt DC to target various CV problems. For example, Zhu *et al.* (Zhu et al. 2021) propose a deformable attention module for object detection. Shim *et al.* (Shim, Park, and Kweon 2020) construct a similarity search and extraction network built upon DC layers for single image super-resolution. Thomas *et al.* (Thomas et al. 2019) introduce a deformable kernel point convolution operator for point clouds. In this paper, we propose a deformable token merging module to adaptively merge more informative image tokens. Unlike previous works that merge tokens from a regular grid, DTM introduces better transformation modeling capability for HVTs.

Proposed Method

Overall Architecture

The overall architecture of LIT is illustrated in Figure 1. Let $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$ be an input RGB image, where H and W represent the height and width, respectively. We first split $\mathcal{I}$ into non-overlapping patches with a patch size of 4×4, and thus the initial feature dimension of each patch is $4 \times 4 \times 3 = 48$. Next, a linear embedding layer is exploited to project each patch into dimension C_1, serving as the initial input for the following pipeline. The entire model is divided into 4 stages. Letting $s \in [1, 2, 3, 4]$ be the index of a stage, we employ L_s blocks at each stage, where the first two stages solely utilise MLP blocks to encode local representations and the last two stages employ standard Transformer blocks (Dosovitskiy et al. 2021) to handle longer dependencies. At each stage, we scale the input feature maps into $\frac{H_{s-1}}{P_s} \times \frac{W_{s-1}}{P_s} \times C_s$, where P_s and C_s represent the patch size and the hidden dimension at the s-th stage, respectively. For the last two stages, we set N_s self-attention heads in each Transformer block.

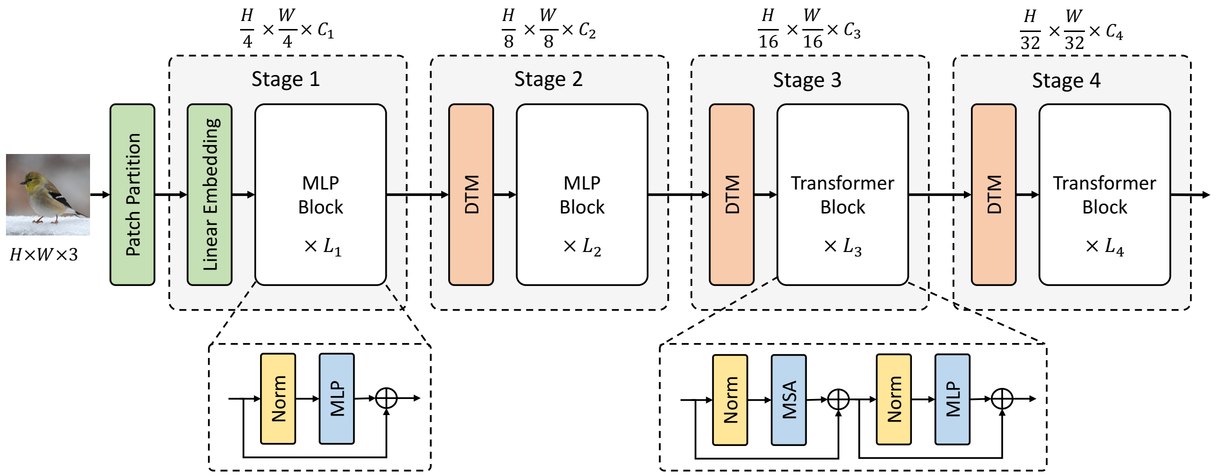
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
Stage 1
Stage 2
Stage 3
Stage 4
$H\times W\times 3$
Patch Partition
Linear Embedding
MLP Block $\times L_1$
DTM
MLP Block $\times L_2$
DTM
Transformer Block $\times L_3$
DTM
Transformer Block $\times L_4$
Norm
MLP
Norm
MSA
Norm
MLP

Figure 1: Overall architecture of LIT. The model is divided into four stages, where we apply MLP blocks in the first two stages and employ standard Transformer blocks in the last two stages. "DTM" denotes our proposed deformable token merging module.

Block Design in LIT

As shown in Figure 1, LIT employs two types of blocks: MLP blocks and Transformer blocks. In the early stages, we apply MLP blocks. Concretely, an MLP block is built upon an MLP which consists of two FC layers with GELU (Hendrycks and Gimpel 2016) non-linearity in between. For each MLP at the s-th stage, an expansion ratio of E_s is used. Specifically, the first FC layer expands the dimension of a token from C_s to $E_s \times C_s$, and the other FC layer reduces the dimension back to C_s. Formally, letting $\mathbf{X} \in \mathbb{R}^{(\frac{H_{s-1}}{P_s} \times \frac{W_{s-1}}{P_s}) \times C_s}$ be the input of the s-th stage and l be the index of a block, an MLP block can be formulated as

$$\mathbf{X}_l = \mathbf{X}_{l-1} + \mathrm{MLP}(\mathrm{LN}(\mathbf{X}_{l-1})), \tag{1}$$

where LN indicates the layer normalization (Ba, Kiros, and Hinton 2016) and MLP denotes an MLP. In the last stages, a Transformer block as described in ViT (Dosovitskiy et al. 2021) contains an MSA layer and an MLP, which can be expressed as

$$\mathbf{X}'_{l-1} = \mathbf{X}_{l-1} + \mathrm{MSA}(\mathrm{LN}(\mathbf{X}_{l-1})), \tag{2}$$
$$\mathbf{X}_l = \mathbf{X}'_{l-1} + \mathrm{MLP}(\mathrm{LN}(\mathbf{X}'_{l-1})). \tag{3}$$

With this architecture, our model benefits from two main advantages: First, we avoid the huge computational costs and memory footprint that are introduced by long sequences in the early stages. Second, unlike recent works that shrink the attention maps using sub-windows (Liu et al. 2021) or reduce the spatial dimensions of the key and value matrices, we keep standard MSA layers in the last two stages so as to maintain the capability of LIT to handle long range dependencies while keeping mild FLOPs due to the pyramid structure. We will show in the ablation study that our simple architecture design outperforms the state-of-the-art hierarchical ViT variants on ImageNet with comparable FLOPs.

Remark. Here we justify the rationality of applying pure MLP blocks in the early stages by considering the relationship among a convolutional layer, an FC layer and an MSA layer. Firstly, we begin with a review of a standard convolutional layer. Let $\mathbf{X} \in \mathbb{R}^{H \times W \times C_{in}}$ be the input feature map, and let $\mathbf{W} \in \mathbb{R}^{K \times K \times C_{in} \times C_{out}}$ be the convolutional weight tensor, where K is the kernel size, C_{in} and C_{out} are the input and output channel dimensions, respectively. For simplicity, we omit the bias term and use $\mathbf{X}_{p,:}$ to represent $\mathbf{X}_{i,j,:,:}$, where (i,j) denotes the pixel index and $p \in [H] \times [W]$. Given a convolutional kernel of $K \times K$ sampling locations, the output for a pixel p can be formulated as

$$\mathrm{Conv}(\mathbf{X})_{p,:} = \sum_{k \in [K \times K]} \mathbf{X}_{p+g(k),:} \mathbf{W}_{g(k),:,:}, \tag{4}$$

where $g : [K \times K] \to \Delta_K$ is a bijective mapping of sampling indexes onto the pre-specified offsets Δ_K. For example, let $\Delta_K = \{(-1,-1), (-1,0), ..., (0,1), (1,1)\}$ be a 3×3 kernel with dilation 1, then $g(0) = (-1,-1)$ represents the first sampling offset.

When $K = 1$, the weight tensor $\mathbf{W}$ is equivalent to a matrix, such that $\mathbf{W} \in \mathbb{R}^{C_{in} \times C_{out}}$. In this case, Eq. (4) can express an FC layer, and the output for a pixel p is defined by

$$\mathrm{FC}(\mathbf{X})_{p,:} = \mathbf{X}_{p,:} \mathbf{W}_{:,:}. \tag{5}$$

Last, let N_h be the number of heads in an MSA layer and $\mathbf{W}^{(h)} \in \mathbb{R}^{C_{in} \times C_{out}}$ be learnable parameters of the h-th head. Under a specific relative positional encoding scheme, Cordonnier *et al.* (Cordonnier, Loukas, and Jaggi 2020) prove that the output from an MSA layer at pixel p can be formulated as

$$\mathrm{MSA}(\mathbf{X})_p = \sum_{h \in [N_h]} \mathbf{X}_{p+f(h),:} \mathbf{W}^{(h)}, \tag{6}$$

where $f : [N_h] \to \Delta_K$ is a bijective mapping of heads onto pixel shifts. In that case, Eq. (6) can be seen as an approximation to a convolutional layer with a kernel size of

$\sqrt{N_h} \times \sqrt{N_h}$. We refer detailed explanations to (Cordonnier, Loukas, and Jaggi 2020).

From Eqs. (4)-(6), we observe that while an MSA layer with sufficient heads is able to approximate any convolutional layer, fewer heads theoretically limit the ability of such approximation. As an extreme case, an MSA layer with one head is only capable of approximating an FC layer. Note that an MSA layer is certainly not equivalent to a convolutional layer in practice. However, d'Ascoli *et al.* (d'Ascoli et al. 2021) observe that the early MSA layers can learn to behave convolutional upon training. Considering most recent HVTs (Wang et al. 2021; Wu et al. 2021) adopt very few heads in the early stages, such convolutional behavior could be limited within small receptive fields. In Figure 3, we show in visualizations that the early MSA layers in PVT-S (Wang et al. 2021) indeed only attend to a tiny area around the query pixel, while removing them costs a minor performance drop but achieves significant reduction in model complexity. This justifies our method of applying pure MLP blocks in the first two stages without self-attention.

Deformable Token Merging

Previous works on HVTs (Wang et al. 2021; Liu et al. 2021) rely on patch merging to achieve pyramid feature representations. However, they merge patches from a regular grid and neglect the fact that not every patch contributes equally to an output unit (Luo et al. 2016). Inspired by deformable convolutions (Dai et al. 2017; Zhu et al. 2019), we propose a deformable token merging module to learn a grid of offsets to adaptively sample more informative patches. Formally, a deformable convolution is formulated as

$$\mathrm{DC}(\mathbf{X})_{p,:} = \sum_{k \in [K \times K]} \mathbf{X}_{p+g(k)+\Delta g(k),:} \mathbf{W}_{g(k),:,:}. \tag{7}$$

Compared to a standard convolution operation as in Eq. (4), DC learns an offset $\Delta g(k)$ for each pre-specified offset $g(k)$. Learning $\Delta g(k)$ requires a separate convolutional layer, which is also applied over the input feature map $\mathbf{X}$. To merge patches in an adaptive manner, we adopt one DC layer in a DTM module, which can be formulated by

$$\mathrm{DTM}(\mathbf{X}) = \mathrm{GELU}(\mathrm{BN}(\mathrm{DC}(\mathbf{X}))), \tag{8}$$

where BN denotes the batch normalization (Ioffe and Szegedy 2015) and we employ the GELU non-linearity. We will show in the ablation study that the sampling locations in DTM are adaptively adjusted when objects' scales and shapes change, benefiting from the learned offsets. Also note that our light-weight DTM introduces negligible FLOPs and parameters compared to regular grid sampling in baselines, thus making it a plug-and-play module for recent HVTs.

Experiments

ImageNet Classification

We conduct experiments on ImageNet (ILSVRC2012) (Russakovsky et al. 2015) dataset. ImageNet is a large-scale dataset which has $\sim$1.2M training images from 1K categories and 50K validation images. We compare with CNN-based ResNet (He et al. 2016) and Transformer-based models including DeiT (Touvron et al. 2021a), PVT (Wang et al. 2021) and Swin (Liu et al. 2021). For simplicity, we denote them as "Model-Ti/S/M/B" to refer to their tiny, small, medium and base variants. Similarly, we define four variants of our LIT models: LIT-Ti, LIT-S, LIT-M and LIT-B. Detailed architecture specifications are included in the supplementary material. For better comparison, we design LIT-Ti as a counterpart to PVT-S, where both models adopt the absolute positional encoding. Our LIT-S, LIT-M and LIT-B use the relative positional encoding, and these three models can be seen as competitors to Swin-Ti, Swin-S, Swin-B, respectively.

Implementation details. In general, all models are trained on ImageNet with 300 epochs and a total batch size of 1024. For all ImageNet experiments, training images are resized to 256×256, and 224×224 patches are randomly cropped from an image or its horizontal flip, with the per-pixel mean subtracted. We use the single-crop setting for testing. We use AdamW optimizer (Loshchilov and Hutter 2019) with a cosine decay learning rate scheduler. The initial learning rate is 1×10^{-3}, and the weight decay is set to 5×10^{-2}. The initial values of learnable offsets in DTM are set to 0, and the initial learning rate for offset parameters is set to 1×10^{-5}. The kernel sizes and strides in DTM are consistent with that of patch merging layers in PVT and Swin. For a fair comparison, we adopt the same training strategies as PVT and Swin when comparing our models to each of them.

Results on ImageNet. In Table 1, we compare LIT with several state-of-the-art methods on ImageNet, ImageNet-Real and ImageNet-V2. In general, all LIT models have fewer parameters, less FLOPs and faster throughput than their counterparts without applying any existing efficient self-attention mechanisms. For memory consumption, at the training time, we observe LIT-Ti and LIT-S require less memory than PVT-S and Swin-Ti while things are on the contrary when comparing LIT-M/B with Swin-S/B. The reason is due to the increased activation memory of standard MSA layers at the later stages. However, at the testing stage, LIT models show advantages over baselines as all of them consume less memory than their counterparts.

In terms of model performance, on ImageNet, LIT-Ti outperforms PVT-S by 1.3% on the Top-1 accuracy while the FLOPs is reduced by 0.2G. LIT-S surpasses Swin-Ti by 0.2% with 0.4G less FLOPs. LIT-M achieves on par performance with Swin-S, whereas the FLOPs of LIT-M is reduced. LIT-B brings 0.1% Top-1 accuracy increase over Swin-B while using 0.4G less FLOPs. For ResNet and DeiT, LIT demonstrates better performance when compared to them with the same magnitude of FLOPs and parameters (*e.g.*, ResNet-50 v.s. LIT-S, DeiT-B v.s. LIT-B). On ImageNet-real, LIT-Ti improves PVT-S by 0.8% on the Top-1 accuracy, while LIT-S/M/B achieve slightly lower accuracy than Swin models. Finally, on ImageNet-V2, LIT models achieve on par or better performance than PVT-S and Swin models. Overall, LIT models present competitive performance across the three datasets, challenging the full self-attention models in recent works.

Method	Params	FLOPs	Throughput	Train Memory	Test Memory	ImageNet@Top-1	Real@Top-1	V2@Top-1
ResNet-18	12M	1.8G	4,454	2,812	1,511	69.8	77.3	57.1
ResNet-50	26M	4.1G	1,279	8,051	2,908	76.2	82.5	63.3
ResNet-101	45M	7.9G	722	10,710	3,053	77.4	83.7	65.7
DeiT-Ti	5M	1.3G	3,398	2,170	314	72.2	80.1	60.4
DeiT-S	22M	4.6G	1,551	4,550	713	79.8	85.7	68.5
DeiT-B	86M	17.5G	582	10,083	1,939	81.8	86.7	71.5
PVT-Ti	13M	1.9G	1,768	4,316	1,269	75.1	82.2	63.0
PVT-S	25M	3.8G	1,007	6,996	1,356	79.8	85.8	68.4
PVT-M	44M	6.7G	680	9,546	1,506	81.2	86.7	70.1
PVT-L	61M	9.8G	481	13,343	1,637	81.7	87.0	71.2
Swin-Ti	28M	4.5G	961	6,211	1,493	81.3	86.6	69.7
Swin-S	50M	8.7G	582	9,957	1,697	83.0	87.6	72.1
Swin-B	88M	15.4G	386	13,705	2,453	83.3	87.7	72.3
LIT-Ti	19M	3.6G	1,294	5,868	1,194	81.1	86.6	70.4
LIT-S	27M	4.1G	1,298	5,973	1,264	81.5	86.4	70.4
LIT-M	48M	8.6G	638	12,248	1,473	83.0	87.3	72.0
LIT-B	86M	15.0G	444	16,766	2,150	83.4	87.6	72.8

Table 1: Comparisons with several state-of-the-art methods on ImageNet (Russakovsky et al. 2015), ImageNet-Real (Beyer et al. 2020) and ImageNet-V2 matched frequency (Recht et al. 2019). All models are trained and evaluated with the input resolution of 224×224. Throughput (imgs/s) is measured on one NVIDIA RTX 3090 GPU, with a batch size of 64 and averaged over 30 runs. Training time and testing time memory consumption is measured with a batch size of 64 in Megabyte (MB).

Model	Params	FLOPs	Top-1 Acc. (%)
PVT-S (Wang et al. 2021)	25M	3.8G	79.8
LIT-Ti*	**19M**	**3.6G**	**80.4**
Swin-Ti (Liu et al. 2021)	28M	4.5G	81.3
LIT-S*	**27M**	**4.1G**	81.3

Table 2: Effect of our architecture design principle. * denotes the LIT model which adopts the same uniform patch merging strategies as in PVT-S or Swin-Ti.

Model	Params	FLOPs	Top-1 Acc. (%)
PVT-S (Wang et al. 2021)	25M	3.8G	79.8
PVT-S + DTM	25M	3.8G	**80.5**
Swin-Ti (Liu et al. 2021)	28M	4.5G	81.3
Swin-Ti + DTM	28M	4.5G	**81.6**

Table 3: Effect of the proposed deformable token merging module. We replace the uniform patch merging strategies in PVT-S and Swin-Ti with our DTM.

Ablation Studies

Effect of the architecture design. To explore the effect of our architecture design principle in LIT, we conduct experiments on ImageNet and compare the architecture of LIT with two recent HVTs: PVT-S (Wang et al. 2021) and Swin-Ti (Liu et al. 2021). The results are shown in Table 2. In general, our architecture improves baseline PVT-S by 0.6% in Top-1 accuracy while using less FLOPs (3.6G v.s. 3.8G). For Swin-Ti, our method reduces the FLOPs by 0.4G while achieving on par performance. It is also worth noting that the total number of parameters is reduced for both PVT-S and Swin-Ti. The overall performance demonstrates the effectiveness of the proposed architecture, which also emphasizes the minor benefits of the early MSAs in PVT and Swin.

Effect of deformable token merging. To verify the effectiveness of our proposed DTM strategy, we replace the default patch merging scheme in PVT-S and Swin-Ti with DTM and train the models on ImageNet. The results are shown in Table 3. We observe that for both models, DTM introduces negligible FLOPs and parameters while improving PVT-S and Swin-Ti by 0.7% and 0.3% in terms of the

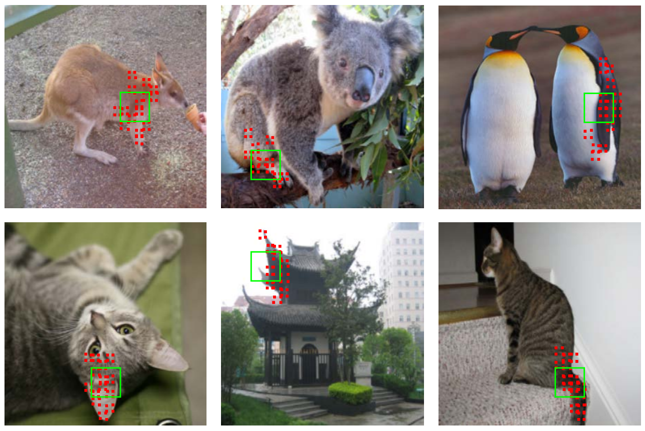

Figure 2: Visualization of learned offsets by the proposed deformable token merging modules. Each image shows 4^3 sampling locations (*red dots*) in three DTM modules with 2×2 filter. The green rectangle outlines a 32×32 patch of the original image, which also indicates a regular sampling field of previous methods. Best viewed in color. More examples can be found in the supplementary material.

Model	Stage	Params	FLOPs	Top-1 (%)
PVT-S	0	25M	3.8G	79.8
PVT-S w/ MSA	0	20M	8.4G	80.9
PVT-S w/ MSA	1	20M	4.5G	80.8
PVT-S w/ MSA	1,2	19M	3.6G	80.4
PVT-S w/ MSA	1,2,3	17M	3.0G	75.0
PVT-S w/ MSA	1,2,3,4	14M	2.8G	66.8

Table 4: Impact of the MSA layers at each stage. Note that PVT-S has four stages, which adopts SRA at all blocks instead of MSA. Here we denote "w/ MSA" as PVT-S with standard MSA layers. "Stage" refers to the stages where we remove all self-attention layers. For example, "1,2" means we remove the self-attention layers in the first two stages. Note that "0" means a model without removing any self-attention layers.

Top-1 accuracy, respectively. Furthermore, we visualize the learned offsets in Figure 2. As it shows, unlike previous uniform patch merging strategy where sampled locations are limited within the green rectangle, our DTM adaptively merges patches according to objects' scales and shapes (*e.g.*, koala leg, cat tail).

Effect of MSA in each stage. To explore the effect of self-attention in recent HVTs, we train PVT-S on ImageNet and gradually remove self-attention layers at each stage. The results are presented in Table 4. First, after replacing the SRA layers in PVT-S with standard MSA layers, we observe 1.1% improvement on the Top-1 accuracy whereas the FLOPs is almost doubled. Next, by gradually removing MSA layers in the first two stages, the Top-1 accuracy only drops by 0.1%, 0.5%, respectively. It implies that the self-attention layers in the early stages of PVT contribute less than expected to the final performance, and they perform not much better than pure MLP layers. It can be attributed to the fact that shallow layers focus more on encoding local patterns. However, we observe a huge performance drop when removing self-attention layers in the last two stages. The results show that the self-attention layers play an important role in the later stages and capturing long range dependencies is essential for well-performed hierarchical vision Transformers.

To better understand the phenomenon, we visualize the attention probabilities for PVT-S without removing any MSA layers, which are depicted in Figure 3. First, the attention map at the first stage shows that the query pixel almost pays no attention to other locations. At the second stage, the receptive field of the query pixel is slightly enlarged, but similar to the first stage. Considering that PVT-S only has one head at the first stage and two heads at the second stage, this strongly supports our hypothesis that very few heads in an MSA layer result in a smaller receptive field, such that a self-attention layer is almost equivalent to an FC layer. Furthermore, we observe relatively larger receptive fields from the attention maps of the last two stages. As a large receptive field usually helps to model longer dependencies, this explains the huge performance drop in Table 4 after we remove the MSA layers in the last two stages.

Model	RetinaNet						
	#P	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
R-50	38M	36.3	55.3	38.6	19.3	40.0	48.8
PVT-S	34M	40.4	61.3	43.0	25.0	42.9	55.7
LIT-Ti	**30M**	**41.6**	**62.8**	**44.7**	**25.7**	**44.4**	**56.4**
R-101	57M	38.5	57.8	41.2	21.4	42.6	51.1
Swin-Ti	39M	41.5	62.1	**44.2**	25.1	**44.9**	55.5
LIT-S	39M	**41.6**	**62.7**	44.1	**25.6**	44.7	**56.5**

Table 5: Object detection performance on the COCO `val2017` split using the RetinaNet framework. "#P" refers to the number of parameters.

Model	Mask R-CNN						
	#P	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
R-50	44M	38.0	58.6	41.4	34.4	55.1	36.7
PVT-S	44M	40.4	62.9	43.8	37.8	60.1	40.3
LIT-Ti	**40M**	**42.0**	**64.9**	**45.6**	**39.1**	**61.9**	**41.9**
R-101	63M	40.4	61.1	44.2	36.4	57.7	38.8
Swin-Ti	48M	42.2	64.6	46.2	39.1	61.6	42.0
LIT-S	48M	**42.9**	**65.6**	**46.9**	**39.6**	**62.3**	**42.4**

Table 6: Object detection and instance segmentation performance on the COCO `val2017` split using the Mask R-CNN framework. AP^b and AP^m denote the bounding box AP and mask AP, respectively. "#P" refers to the number of parameters.

Object Detection and Instance Segmentation on COCO

In this section, we conduct experiments on COCO 2017 (Lin et al. 2014) dataset to show the performance of LIT on object detection and instance segmentation. COCO is a large-scale dataset which contains $\sim$118K images for the training set and $\sim$5K images for the validation set. For a fair comparison, we evaluate LIT-Ti and LIT-S on two base detectors: RetinaNet (Lin et al. 2017b) and Mask R-CNN (He et al. 2017). For the experiments with both detectors, we consider CNN-based ResNet (He et al. 2016) and Transformer-based models including PVT-S (Wang et al. 2021) and Swin-Ti (Liu et al. 2021). Following common practice (Carion et al. 2020; Wang et al. 2021), we measure the performance of all models by Average Precision (AP) in COCO.

Implementation details. All models are trained on 8 V100 GPUs, with $1\times$ schedule (12 epochs) and a total batch size of 16. We use AdamW (Loshchilov and Hutter 2019) optimizer with a step decay learning rate scheduler. Following PVT (Wang et al. 2021), the initial learning rates are set to 1×10^{-4} and 2×10^{-4} for RetinaNet and Mask R-CNN, respectively. The weight decay is set to 1×10^{-4} for all models. Results of Swin-Ti based detectors are adopted from Chu *et al.* (Chu et al. 2021). At the training stage, we initialize the backbone with the pretrained weights on ImageNet. The training images are resized to the shorter size of 800 pixels, and the longer size is at most 1333 pixels. During inference, we fix the shorter side of an image to 800 pixels.

Results on COCO. Table 5 shows the comparisons of dif-

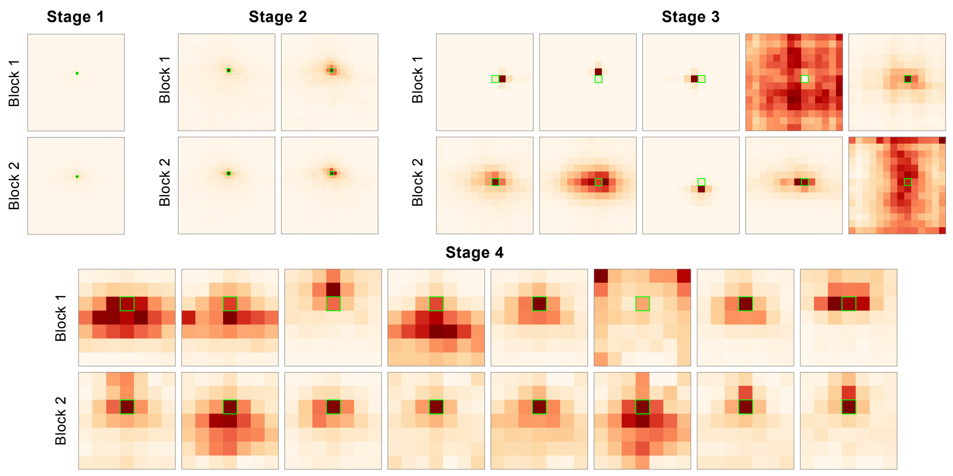

Figure 3: Attention probabilities of PVT-S with standard MSA layers. For each stage, we visualize the attention map of each head (*columns*) at selected blocks (*rows*). All attention maps are averaged over 100 validation images. Each map shows the attention probabilities of a query pixel (*green rectangle*) to other pixels. Darker color indicates higher attention probability and vice versa. Best viewed in color. We provide visualizations of all blocks in the supplementary material.

ferent backbones on object detection based on RetinaNet. By comparing LIT with PVT and ResNet counterparts, we find that our model outperforms both backbones on object detection in almost all metrics. Similar results can be found in Table 6, where LIT again surpasses compared methods on object detection and instance segmentation using the Mask R-CNN framework.

Semantic Segmentation on ADE20K

We conduct experiments on ADE20K (Zhou et al. 2019) to show the performance of LIT models on semantic segmentation. ADE20K is a widely adopted dataset for semantic segmentation, which has ~20K training images, ~2K validation images and ~3K test images. For a fair comparison, we evaluate LIT models with Semantic FPN (Kirillov et al. 2019). Following the common practice in (Wang et al. 2021), we measure the model performance by mIoU.

Implementation details. All models are trained on 8 V100 GPUs, with 8K steps and a total batch size of 16. The AdamW (Loshchilov and Hutter 2019) optimizer is adopted with an initial learning rate of 1×10^{-4}. Learning rate is decayed by the polynomial decay schedule with the power of 0.9. We set the weight decay to 1×10^{-4}. All backbones are initialized with the pretrained weights on ImageNet. At the training stage, we randomly resize and crop the images to 512×512. During inference, images are scaled to the short size of 512.

Results on ADE20K. We compare different backbones on the ADE20K validation set in Table 7. From the results, we observe that LIT-Ti outperforms ResNet-50 and PVT-S by 4.6% and 1.5% mIoU, respectively. For LIT-S, our model again surpasses ResNet-101 and Swin-Ti, with 2.9% and 0.2% improvement on mIoU, respectively. The overall performance demonstrates the effectiveness of the proposed LIT models for dense prediction tasks.

Backbone	Semantic FPN	
	Params (M)	mIoU (%)
ResNet-50	29	36.7
PVT-S	28	39.8
LIT-Ti	24	**41.3**
ResNet-101	48	38.8
Swin-Ti	32	41.5
LIT-S	32	**41.7**

Table 7: Semantic segmentation performance with different backbones on the ADE20K validation set.

Conclusion and Future Work

In this paper, we have introduced LIT, a hierarchical vision transformer which pays less attention in the early stages to ease the huge computational cost of self-attention modules over high-resolution representations. Specifically, LIT applies MLP blocks in the first two stages to focus on local patterns while employing standard Transformer blocks with sufficient heads in the later stages to handle long range dependencies. Moreover, we have proposed a deformable token merging module, which is learned to adaptively merge informative patches to an output unit, with enhanced geometric transformations. Extensive experiments on ImageNet, COCO and ADE20K have demonstrated that LIT achieves better performance compared with existing state-of-the-art HVT methods. Future works may include finding better architectural configurations of LIT with neural architecture search (NAS) and improving MLP blocks in the early stages to enhance the capability of LIT to encode local patterns. Besides, one may also consider applying efficient self-attention mechanisms (Peng et al. 2021; Wang et al. 2020) in the later stages to achieve better efficiency.

References

Ba, J.; Kiros, J.; and Hinton, G. E. 2016. Layer Normalization. *ArXiv*, abs/1607.06450.

Beyer, L.; Hénaff, O. J.; Kolesnikov, A.; Zhai, X.; and van den Oord, A. 2020. Are we done with ImageNet? *arXiv preprint arXiv:2006.07159*.

Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; and Zagoruyko, S. 2020. End-to-End Object Detection with Transformers. In *ECCV*, 213–229.

Chen, B.; Li, P.; Li, C.; Li, B.; Bai, L.; Lin, C.; Sun, M.; Yan, J.; and Ouyang, W. 2021a. GLiT: Neural Architecture Search for Global and Local Image Transformer. In *ICCV*.

Chen, M.; Peng, H.; Fu, J.; and Ling, H. 2021b. AutoFormer: Searching Transformers for Visual Recognition. In *ICCV*.

Chu, X.; Tian, Z.; Wang, Y.; Zhang, B.; Ren, H.; Wei, X.; Xia, H.; and Shen, C. 2021. Twins: Revisiting the Design of Spatial Attention in Vision Transformers. *Arxiv preprint 2104.13840*.

Cordonnier, J.; Loukas, A.; and Jaggi, M. 2020. On the Relationship between Self-Attention and Convolutional Layers. In *ICLR*.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable Convolutional Networks. In *ICCV*, 764–773.

d'Ascoli, S.; Touvron, H.; Leavitt, M. L.; Morcos, A. S.; Biroli, G.; and Sagun, L. 2021. ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases. In *ICML*, 2286–2296.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *ACL*, 4171–4186.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR*.

Graham, B.; El-Nouby, A.; Touvron, H.; Stock, P.; Joulin, A.; Jégou, H.; and Douze, M. 2021. LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference. *arXiv preprint arXiv:22104.01136*.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. B. 2017. Mask R-CNN. In *ICCV*, 2980–2988.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian Error Linear Units (GELUs). *arXiv: Learning*.

Heo, B.; Yun, S.; Han, D.; Chun, S.; Choe, J.; and Oh, S. J. 2021. Rethinking spatial dimensions of vision transformers. In *ICCV*.

Hou, Q.; Cheng, M.; Hu, X.; Borji, A.; Tu, Z.; and Torr, P. H. S. 2019. Deeply Supervised Salient Object Detection with Short Connections. *TPAMI*, 41(4): 815–828.

Hu, H.; Zhang, Z.; Xie, Z.; and Lin, S. 2019. Local Relation Networks for Image Recognition. In *ICCV*, 3463–3472.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 448–456.

Kirillov, A.; Girshick, R. B.; He, K.; and Dollár, P. 2019. Panoptic Feature Pyramid Networks. In *CVPR*, 6399–6408.

Li, Y.; Zhang, K.; Cao, J.; Timofte, R.; and Van Gool, L. 2021. LocalViT: Bringing Locality to Vision Transformers. *arXiv preprint arXiv:2104.05707*.

Lin, T.; Dollár, P.; Girshick, R. B.; He, K.; Hariharan, B.; and Belongie, S. J. 2017a. Feature Pyramid Networks for Object Detection. In *CVPR*, 936–944.

Lin, T.; Goyal, P.; Girshick, R. B.; He, K.; and Dollár, P. 2017b. Focal Loss for Dense Object Detection. In *ICCV*, 2999–3007.

Lin, T.; Maire, M.; Belongie, S. J.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: Common Objects in Context. In Fleet, D. J.; Pajdla, T.; Schiele, B.; and Tuytelaars, T., eds., *ECCV*, 740–755.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled weight decay regularization. In *ICLR*.

Luo, W.; Li, Y.; Urtasun, R.; and Zemel, R. S. 2016. Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. In *NeurIPS*, 4898–4906.

Pan, Z.; Zhuang, B.; Liu, J.; He, H.; and Cai, J. 2021. Scalable visual transformers with hierarchical pooling. In *ICCV*.

Parmar, N.; Ramachandran, P.; Vaswani, A.; Bello, I.; Levskaya, A.; and Shlens, J. 2019. Stand-Alone Self-Attention in Vision Models. In *NeurIPS*, 68–80.

Peng, H.; Pappas, N.; Yogatama, D.; Schwartz, R.; Smith, N. A.; and Kong, L. 2021. Random feature attention. In *ICLR*.

Recht, B.; Roelofs, R.; Schmidt, L.; and Shankar, V. 2019. Do ImageNet Classifiers Generalize to ImageNet? In *ICML*, 5389–5400.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV*, 211–252.

Shim, G.; Park, J.; and Kweon, I. S. 2020. Robust Reference-Based Super-Resolution With Similarity-Aware Deformable Convolution. In *CVPR*, 8422–8431.

Srinivas, A.; Lin, T.-Y.; Parmar, N.; Shlens, J.; Abbeel, P.; and Vaswani, A. 2021. Bottleneck transformers for visual recognition. In *CVPR*.

Tenney, I.; Das, D.; and Pavlick, E. 2019. BERT Rediscovers the Classical NLP Pipeline. In Korhonen, A.; Traum, D. R.; and Màrquez, L., eds., *ACL*, 4593–4601.

Thomas, H.; Qi, C. R.; Deschaud, J.; Marcotegui, B.; Goulette, F.; and Guibas, L. J. 2019. KPConv: Flexible and Deformable Convolution for Point Clouds. In *ICCV*, 6410–6419.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021a. Training data-efficient image transformers & distillation through attention. In *ICML*, 10347–10357.

Touvron, H.; Cord, M.; Sablayrolles, A.; Synnaeve, G.; and Jégou, H. 2021b. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*.

Vaswani, A.; Ramachandran, P.; Srinivas, A.; Parmar, N.; Hechtman, B.; and Shlens, J. 2021. Scaling Local Self-Attention For Parameter Efficient Visual Backbones. In *CVPR*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NeurIPS*, 5998–6008.

Wang, S.; Li, B.; Khabsa, M.; Fang, H.; and Ma, H. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*.

Wu, H.; Xiao, B.; Codella, N.; Liu, M.; Dai, X.; Yuan, L.; and Zhang, L. 2021. Cvt: Introducing convolutions to vision transformers. In *ICCV*.

Wu, Z.; Su, L.; and Huang, Q. 2019. Cascaded Partial Decoder for Fast and Accurate Salient Object Detection. In *CVPR*, 3907–3916.

Yan, H.; Li, Z.; Li, W.; Wang, C.; Wu, M.; and Zhang, C. 2021. ConTNet: Why not use convolution and transformer at the same time? *arXiv preprint arXiv:2104.13497*.

Yuan, K.; Guo, S.; Liu, Z.; Zhou, A.; Yu, F.; and Wu, W. 2021a. Incorporating Convolution Designs into Visual Transformers. *arXiv preprint arXiv:2103.11816*.

Yuan, L.; Chen, Y.; Wang, T.; Yu, W.; Shi, Y.; Tay, F. E.; Feng, J.; and Yan, S. 2021b. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*.

Zhou, B.; Zhao, H.; Puig, X.; Xiao, T.; Fidler, S.; Barriuso, A.; and Torralba, A. 2019. Semantic Understanding of Scenes Through the ADE20K Dataset. *IJCV*, 302–321.

Zhou, D.; Kang, B.; Jin, X.; Yang, L.; Lian, X.; Hou, Q.; and Feng, J. 2021. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*.

Zhu, X.; Hu, H.; Lin, S.; and Dai, J. 2019. Deformable ConvNets V2: More Deformable, Better Results. In *CVPR*, 9308–9316.

Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; and Dai, J. 2021. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In *ICLR*.