Super Vision Transformer

Mingbao Lin[1,2‡] **Mengzhao Chen**[1‡] **Yuxin Zhang**[1] **Ke Li**[2]
Yunhang Shen[2] **Chunhua Shen**[3] **Rongrong Ji**[1*]

[1]MAC Lab, School of Informatics, Xiamen University, China
[2]Tencent Youtu Lab [3]Zhejiang University, China

`lmb001@outlook.com, {cmzxmu,yuxinzhang}@stu.xmu.edu.cn`
`{tristanli.sh,shenyunhang01,chhshen}@gmail.com, rrji@xmu.edu.cn`

Abstract

We attempt to reduce the computational costs in vision transformers (ViTs), which increase quadratically in the token number. We present a novel training paradigm that trains only one ViT model at a time, but is capable of providing improved image recognition performance with various computational costs. Here, the trained ViT model, termed super vision transformer (SuperViT), is empowered with the versatile ability to solve incoming patches of multiple sizes as well as preserve informative tokens with multiple keeping rates (the ratio of keeping tokens) to achieve good hardware efficiency for inference, given that the available hardware resources often change from time to time. Experimental results on ImageNet demonstrate that our SuperViT can considerably reduce the computational costs of ViT models with even performance increase. For example, we reduce 2× FLOPs of DeiT-S while increasing the Top-1 accuracy by 0.2% and 0.7% for 1.5× reduction. Also, our SuperViT significantly outperforms existing studies on efficient vision transformers. For example, when consuming the same amount of FLOPs, our SuperViT surpasses the recent state-of-the-art (SoTA) EViT by 1.1% when using DeiT-S as their backbones. The project of this work is made publicly available at `https://github.com/lmbxmu/SuperViT`.

1 Introduction

Vision transformers (ViTs) initially introduced in 2020 [13] have spread widely in the field of computer vision and soon become one of the most pervasive and promising architectures in varieties of prevalent vision tasks, such as image classification [13, 23, 14], object detection [4, 52], video understanding [2, 1] and many others [51, 41, 26, 49, 21]. The basic idea behind ViTs is to break down an image as a series of local patches and use a linear projection to tokenize these patches as inputs. In particular, ViTs merit in its property of capturing the long-range relationships between different portions of an image with the mechanism of multi-head self-attention (MHSA).

Recent studies focus more on an efficient ViT [28, 9, 25, 14, 5] since the excessive computational costs, which increase quadratically to the number of tokens, have severely barricaded the broader usage of ViTs in real-world applications. Note that the transformer's token sequence length is inversely proportional to the square of the patch size, which denotes that models with smaller patch sizes are computationally more expensive. The most intuitive way is to reduce the transformer's token number by enlarging the patch size. However, it has been an experimental consensus in the literature [13] that a ViT model performs better with smaller-size patches as its inputs. For example, ViT-B [13] observes 77.91% Top-1 accuracy on ImageNet when the patch size is 16×16 while only

[*]Corresponding author; [‡]Equal contributions.

Preprint. Under review.

73.38% is reached if the patch size is 32×32. Modern ViT structures simply accept a fixed patch size *w.r.t.* all input images when training a ViT model. It remains unexplored to train ViT models with a larger patch size for an ecomonic computation while injecting information of a smaller patch size to retain the performance.

Luckily, images are often filled with redundant regions, such as backgrounds. This property has inspired many researchers to go further, and drop the less informative tokens after forwarding the token sequence to the networks. Tang *et al.* [34] introduced a top-down token pruning paradigm. DynamicViT [31] and IA-RED^2 [30] learn to score each token with a learnable prediction module, while EViT [27] utilizes off-the-shelf class attention to measure token importance. In DVT [39], multiple ViTs are cascaded and each image's preserved token number is decided by an early-existing policy. Despite that these token drop methods decrease computation costs, they sacrifice recognition accuracy. For example, the recent PS-ViT [34] and Evo-ViT [27] decrease the Top-1 performance of DeiT-S [35] by 0.4% although 1.6∼2.0G FLOPs are saved on ImageNet. Most existing methods train a ViT model on the premise of a fixed token keeping rate. It remains an open issue to train one ViT under multiple token keeping rates such that the case of more token drops for computation savings can benefit performance increase from that of fewer drops.

Above all, most existing methods are restricted to processing token sequence with a permanent patch size or excavating token redundancy with a fixed keeping rate. Once the training is finished, the inference process is deterministic, thus these methods result in a trained model with a static complexity, which not only bears a poor trade-off between performance and inference cost, but fails to support a good hardware efficiency given that the hardware, even on the same workstation, is often equipped with different battery conditions or workloads at different periods.

We present a novel training paradigm that derives only one ViT model at a time but is endowed with a versatile ability of image recognition and its complexity can dynamically adapt to the current hardware resources. Figure 1 illustrates our training framework. We make copies of an input image into multiple parallel branches each of which is split into local patches of a particular size. These patch branches are sequentially fed to a ViT model to take in multi-size patch information. For each patch sequence, we also make efforts to excavate redundant regions from the perspective of training a network with multiple token keeping rates. Consequently, to our best knowledge, this is the first study that can obtain one ViT model of recognizing images at multiple complexities in inference. The trained ViT model in this paper, is termed as super vision transformer (SuperViT), where "super" refers to the ability to dispose of incoming patches of different sizes as well as preserving informative tokens with varying keeping rates. When compared with the very recent token drop methods [31, 27, 30, 43], our SuperViT has the following advantages: (1) In contrast to these studies on token drops that sacrifice recognition accuracy, we observe that our training paradigm provides a better recognition ability. For example, the backbone network of DeiT-S increases by 0.2% on ImageNet even when 50% tokens are removed. (2) Our SuperViT provides better hardware efficiency since it allows a fast accuracy-efficiency trade-off by adapting the patch size of input images as well as token keeping rate to fitting the currently available hardware resources.

2 Related Work

The pioneering ViT work dates back to [13] that applies a pure transformer in natural language processing [37] to image classification. However, due to the lack of inductive bias, its state-of-the-art performance depends too much on a very large-scale data corpus such as JFT-300M [33]. To overcome the necessity of large datasets for training the transformer, DeiT [35] introduces token-based distillation and strong data augmentation. Since then, based on the main spirit of ViTs, substantial breakthroughs have been made in computer vision society. Except for the studies on token drops discussed in Sec. 1, we further briefly revisit some related studies below and also encourage the readers to go further on the survey paper [24, 16] for a more comprehensive overview.

Albeit ViT's advantage in capturing long-range relations between image patches, it fails to model local patch information. This motivates the researchers to compute self-attention within the local/windowed region. Swin Transformer [28] devises a shifted window partitioning to realize cross-window connections while the attention is performed upon each local window. TNT [17] leverages an inner block to strengthen the interactions of pixel information within each patch. Twins [9] alternates locally-grouped self-attention and global sub-sampled attention layer-by-layer. These works not only bring back local information but also improve the efficiency of ViT models since the whole computation of global attention is avoidable.

Apart from the above studies, there are also many variants to follow the footprints of convolutional neural networks (CNNs). Shuffle Transformer [22] implements information flow among windows by a spatial shuffle operation as in ShuffleNet [50]. Alike to the depthwise convolution and pointwise convolution in MobileNet [19], SepViT [25] devises depthwise self-attention to capture local representation within each window and pointwise self-attention to build connections among windows. By incorporating the pyramid structure from CNNs, PVT [38] serves as a versatile backbone for many dense prediction tasks. DeformableViT [40] equips with a deformable self-attention module in line with deformable convolution [11] to enable flexible spatial locations conditioned on input data. ViT-Slim [5] searches for a sub-transformer network across three dimensions of input tokens, MHSA and MLP modules with a ℓ_1-regularized soft mask to indicate the global importance of dimensions, just like the CNNs network slimming [29].

Also, constructing the transformer with convolutions can be a more straightforward way to solve the inductive bias. CPVT [10] uses a convolution layer to replace the learnable positional embedding for fine-level feature encoding. CMT [15] hybridizes CNNs and transformers to respectively capture local and global information, which promotes the ability of network representation. Graham *et al.* [14] revisited principles from extensive CNN studies. Consequently, LeViT, a transformer architecture inspired by convolutional approaches, is proposed with a trade-off between accuracy performance and inference speeds. Mobile-Former [8] parallelizes MobileNets [19] and transformers with a two-way bridge to fuse local and global features.

3 Methodology

3.1 Overview

Vision transformer (ViT) breaks down an image $I \in \mathbb{R}^{H \times W \times C}$ into a set of N local patches with shape of $P \times P \times C$. We have $N = {(H \cdot W)}/{(P \cdot P)}$. These patches are linearly projected into D-dimensional token vectors. Also, an extra [class] token, which learns global image information and is responsible for the final classification, is added to the token sets. As results, the input token sequence of a ViT model can be represented as:

$$\mathbf{X}^0 = [\mathbf{x}^0_{cls}; \mathbf{x}^0_1; ...; \mathbf{x}^0_N] + \mathbf{E}_{pos}, \tag{1}$$

where $\mathbf{x}^0_{cls} \in \mathbb{R}^D$ denotes [class] token and $\mathbf{x}^0_i$ represents the token of the i-th patch with $i > 0$. The $\mathbf{E}_{pos}$ denotes the learnable position embedding. Then, all tokens are fed into a ViT model $\mathcal{T}$ with L sequentially-stacked transformer encoders, each of which consists of a multi-head self-attention (MHSA) layer and a feed-forward network (FFN). Denoting $\mathbf{X}^{l-1}$ and $\mathbf{X}^l$ as the input and output of the l-th transformer encoder, the processing of MHSA and FFN is formulated as:[2]

$$\mathbf{Y}^l = \mathbf{X}^{l-1} + \text{MHSA}(\mathbf{X}^{l-1}), \tag{2}$$

$$\mathbf{X}^l = \mathbf{Y}^l + \text{FFN}(\mathbf{Y}^l). \tag{3}$$

Usually, FFN consists of two fully-connected layers with a non-linear mapping inserted in-between such as GELU [18]. In MHSA, the input tokens are linearly mapped to three matrices including a query $\mathbf{Q}$, a key $\mathbf{K}$ and a value $\mathbf{V}$, and the MHSA can be formulated as:

$$\text{MHSA}(\mathbf{X}^{l-1}) = \text{Concat}\Big[\text{Attention}(\mathbf{Q}^{l,h}, \mathbf{K}^{l,h})\mathbf{V}^{l,h}\Big]_{h=1}^{H} \mathbf{W}^l, \tag{4}$$

where Concat[$\cdot$] concatenates its inputs and we have $\mathbf{Q}^l = \text{Concat}[\mathbf{Q}^{l,h}]_{h=1}^H$, $\mathbf{K}^l = \text{Concat}[\mathbf{K}^{l,h}]_{h=1}^H$, $\mathbf{V}^l = \text{Concat}[\mathbf{V}^{l,h}]_{h=1}^H$. $\mathbf{W}^l$ is a projection matrix. Attention$(\cdot,\cdot)$ is computed as:

$$\text{Attention}(\mathbf{Q}^{l,h}, \mathbf{K}^{l,h}) = [\mathbf{a}^{l,h}_{cls}; \mathbf{a}^{l,h}_1; ...; \mathbf{a}^{l,h}_N] = \text{Softmax}\big(\frac{\mathbf{Q}^{l,h}(\mathbf{K}^{l,h})^T}{\sqrt{D}}\big). \tag{5}$$

In particular, $\mathbf{a}^l_{cls} = \sum_{h=1}^H \mathbf{a}^{l,h}_{cls}$ is known as the attention from [class] token to all patch tokens and often used to determine the information richness of each token [27, 43, 7]. After a series of

[2]Layer normalization is usually inserted before MHSA and FFN. We omit it here for brevity.

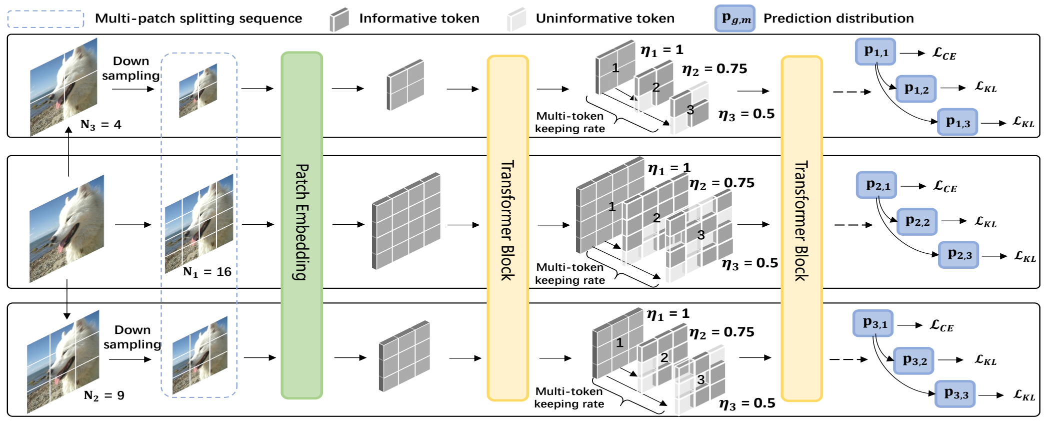

Multi-patch splitting sequence
Informative token
Uninformative token
$\mathbf{p}_{g,m}$ Prediction distribution
Down sampling
$N_3 = 4$
$N_1 = 16$
$N_2 = 9$
Patch Embedding
Transformer Block
$\eta_1 = 1$
$\eta_2 = 0.75$
$\eta_3 = 0.5$
Multi-token keeping rate
$\mathbf{p}_{1,1}$ $\mathcal{L}_{CE}$
$\mathbf{p}_{1,2}$ $\mathcal{L}_{KL}$
$\mathbf{p}_{1,3}$ $\mathcal{L}_{KL}$
$\mathbf{p}_{2,1}$ $\mathcal{L}_{CE}$
$\mathbf{p}_{2,2}$ $\mathcal{L}_{KL}$
$\mathbf{p}_{2,3}$ $\mathcal{L}_{KL}$
$\mathbf{p}_{3,1}$ $\mathcal{L}_{CE}$
$\mathbf{p}_{3,2}$ $\mathcal{L}_{KL}$
$\mathbf{p}_{3,3}$ $\mathcal{L}_{KL}$

Figure 1: Training framework of SuperViT. An image is arranged into multiple branches of different patch sizes. For each branch, various token keeping rates are considered in ViT training. Thus, the trained SuperViT is capable of better recognizing images with different computational costs.

MHSA-FFN transformation, the [class] token $\mathbf{x}_L^{cls}$ is extracted from the L-th transformer encoder and utilized for object category prediction. Taking classification as an example, $\mathbf{x}_L^{cls}$ is fed to a classifier consisting of a fully-connected layer and a softmax layer. Therefore, given an L-layer ViT model $\mathcal{T}$ with the input token sequence $\mathbf{X}_0 \in \mathbb{R}^{N \times D}$, the category prediction distribution is obtained as:

$$\mathbf{p} = \mathcal{T}(\mathbf{X}^0) = \text{Softmax}\big(\text{FC}(\mathbf{x}_{cls}^L)\big). \tag{6}$$

Discussion. The ViT model benefits from the MHSA that models the long-range dependencies between the input tokens. However, the main computational complexity also stems from the MHSA layer [13, 28] as $\mathcal{O}(\text{MHSA}) = 4ND^2 + 2N^2D$. We can see that the complexity of MHSA increases quadratically in the number of incoming tokens N. Consequently, the MHSA has become the computation bottleneck in the ViT model. The first naive manner is to reduce the transformer's input sequence length by enlarging the size of split patches. However, as analyzed in Sec. 1, the performance of a ViT model is closely correlated with the patch size as well. The second intuitive approach is to discard tokens considered less informative in the network forward, which is observed to deteriorate the accuracy as discussed in Sec. 1.

We analyze that existing methods [31, 27, 23, 35] are overfitting to a static complexity since they process token sequence with a permanent length or excavating token redundancy with a fixed keeping rate. The resulting ViT models suffer a poor accuracy-speed trade-off, as well as fail to support good hardware efficiency. This impels us to learn one versatile transformer of recognizing images at multiple complexities, named super vision transformer (SuperViT), details of which are given below.

3.2 Super Vision Transformer

In this section, we aim to train only one ViT model that is able to maintain computational costs at different levels. We present how our SuperViT is empowered with a better capability of image recognition and can adapt to the current availability of hardware resources. A pipeline overview of our SuperViT training is depicted in Figure 1, which mainly includes a part of multi-size patch splitting and a part of multi-token keeping rate. More are elaborated below.

3.2.1 Multi-Size Patch Splitting

The contradiction exists that larger-size patches reduces the computational costs while the recognition benefits from smaller-size patches. To maintain good performance at low computation, we propose to inject information of larger-size patches into the training of a ViT model with a smaller-size input. That is, we intend to equip SuperViT with the ability to solve incoming patches of multiple sizes.

To that end, as shown in Figure 1, the input image is copied into G parallel branches and each branch is responsible for a particular patch size in the image splitting. Consequently, we have a patch size set $\{P_g \times P_g\}_{g=1}^G$ where $P_g > P_{g+1}$. Then, an input image $I \in \mathbb{R}^{H \times W \times C}$ is split into G patch sets and the g-th set consists of local patches with a shape of $P_g \times P_g \times C$, leading to a sequence length of $N_g = \frac{H \cdot W}{P_g \cdot P_g}$. Modern ViT structures simply accept a fixed size of input patches *w.r.t.* all training images such that the patch sequence can be embedded into tokens of the same dimension for training a ViT model in parallel. To embed patches of different shapes into the token vectors in a D-dimensional space, a simple approach is to consider individual patch embedding layers for each patch set [53, 39], which however, increases the parameters. Instead, we choose the economical bilinear interpolation to downsample/upsample these local patches for a shape alignment first. Then, these aligned patches are fed to a shared embedding layer to obtain the input token sequence $\mathbf{X}_g^0$ for the g-th patch set where $\mathbf{X}_g^0$ is defined as:

$$\mathbf{X}_g^0 = \left[\mathbf{x}_{g,cls}^0; \mathbf{x}_{g,1}^0; ...; \mathbf{x}_{g,N_g}^0\right] + \mathbf{E}_{g,pos}, \tag{7}$$

where $\mathbf{x}_{g,cls}^0$ and $\mathbf{x}_{g,i}^0 (i > 0)$ represent the class token and the i-th token in the g-th input patch set. Then, we feed the token sequence $\{\mathbf{X}_g^0\}_{g=1}^G$ to the ViT one-by-one for a series of MHSA-FFN transformations described in Sec. 3.1. Finally, we obtain a distribution set of category predictions $\{\mathbf{p}_g\}_{g=1}^G$, where each prediction $\mathbf{p}_g$ is derived by:

$$\mathbf{p}_g = \mathcal{T}(\mathbf{X}_g^0) = \text{Softmax}\big(\text{FC}(\mathbf{x}_{g,cls}^L)\big). \tag{8}$$

Then, the prediction distributions $\{\mathbf{p}_g\}_{g=1}^G$ can be used to formulate the learning objective, such as cross-entropy loss with the ground-truth labels for ViT training. The trade-off between computational budget and accuracy performance can hardly be made if a ViT model is trained under a single patch size such as $P_1 \times P_1$. Luckily, our SuperViT can well enhance the performance of $P_1 \times P_1$ at test stage since information of smaller-size patches is injected during network training. Moreover, the performance of smaller-size patches can also be enhanced by larger-size ones since images of different complexities require different patch sizes to be correctly classified [39, 7].

3.2.2 Multi-Token Keeping Rate

We further spend efforts to reduce computational costs of our SuperViT given that redundant regions widely exist in image contents. As discussed in Sec. 1, most existing studies on dropping less informative tokens suffer poor performance since they train the ViT model with a fixed token keeping rate, which fails to adapt to images of different complexities. Therefore, we propose to strengthen our SuperViT with the ability to preserve informative tokens with multiple keeping rates.

To that effect, as shown in Figure 1, we predesignate a set of token keeping rates $\{\eta_m\}_{m=1}^M$ where $\eta_m > \eta_{m+1}$, and $0 < \eta_m < 1$ for $m > 1$ which means the top-$(\eta_m \cdot N)$ informative tokens from the input sequence will be preserved. We define $\eta_1 = 1$, which indicates the case of preserving all tokens is always performed. Then, for token sequence in the l-th layer $\mathbf{X}_g^l = [\mathbf{x}_{g,cls}^l; \mathbf{x}_{g,1}^l; ...; \mathbf{x}_{g,N_g}^l]$, we determine the information richness of each token according to the class attention $\mathbf{a}_{cls}^l$.

As defined in Eq. (5), the i-th entry of $\mathbf{a}_{cls}^l$ determines how much information of the i-th token $\mathbf{x}_{g,i}^l$ is fused into the class token $\mathbf{x}_{g,cls}^l$ [27]. It thus has become an indicator to reflect the information richness of each token in many existing studies [27, 43, 7]. We focus more on training one versatile ViT model proficient in processing multi-token keeping rates, thus following existing studies, we directly preserve tokens with larger attention values in this paper.

With the predefined keeping rate set $\{\eta_m\}_{m=1}^M$, we sequentially feed the incoming token sequence for MHSA-FFN transformations. The m-th forward propagation is constrained by token keeping rate η_m and the resulting category prediction distribution is formulated as:

$$\mathbf{p}_{g,m} = \mathcal{T}(\mathbf{X}_g^0 | \eta_m) = \text{Softmax}\big(\text{FC}(\mathbf{x}_{g,cls}^L | \eta_m)\big), \tag{9}$$

In particular, for $\eta_1 = 1$, we have $\mathbf{p}_{g,1} = \mathbf{p}_g = \mathcal{T}(\mathbf{X}_g^0)$. Similar to the analysis for multi-size patch splitting, training a ViT model with multiple token keeping rates also benefits the performance while the less informative tokens are removed for computation savings.

3.3 Training Objective

Based on the proposed multi-size patch splitting and multi-token keeping rate, our SuperViT would result in a total of $G \times M$ different computational costs. Inspired by the one-shot training settings in traditional CNNs [47, 3, 45], in each training iteration, we first freeze updating our SuperViT and sequentially forward a token sequence with a particular patch size as well as token keeping rate to derive the corresponding category prediction distribution set $\{\mathbf{p}_{g,m}\}_{g=1:G,m=1:M}$. Among these predictions, it is expected that $\mathbf{p}_{g,1}$ can fit best with the ground-truth label $\mathbf{y}$ since no token drop is performed in this case. Thus, we propose to supervise the learning of SuperViT with the ground-truth labels in the case of no performing token drop, and use $\mathbf{p}_{g,1}$ as a knowledge hint to guide the learning of SuperViT in the case of performing token drops, leading to our training objective as:

$$\mathcal{L} = \sum_{g=1}^{G} \text{CE}(\mathbf{p}_{g,1}, \mathbf{y}) + \sum_{g=1}^{G} \sum_{m=2}^{M} \text{KL}(\mathbf{p}_{g,m}, \mathbf{p}_{g,1}), \tag{10}$$

where CE$(\cdot, \cdot)$ denotes the cross-entropy loss and KL$(\cdot, \cdot)$ represents the Kullback-Leibler divergence. After calculating the training loss, we switch on the gradient computing to update our SuperViT. However, looping over all the $G \cdot M$ cases causes heavy training burden. Instead, we offer an alternative where only four complexities are considered in each iteration to reduce the training consumption. The (patch size, keeping rate) tuple of these four complexities includes ($P_G \times P_G$, η_1), ($P_G \times P_G$, η_{m_1}), ($P_{g_1} \times P_{g_1}$, η_1), ($P_{g_1} \times P_{g_1}$, η_{m_2}) where η_{m_1} and η_{m_2} are randomly sampled from $\{\eta_m\}_{m=2}^{M}$ and $P_{g_1} \times P_{g_1}$ is randomly sampled from $\{P_g \times P_g\}_{g=1}^{G-1}$.

Hardware Efficiency. Our training paradigm results in one single ViT model executable to recognize images at different computational costs. It can be well deployed upon various hardware platforms with different resource constraints. Even for a given hardware device, our SuperViT permits instant and adaptive accuracy-efficiency trade-offs at runtime once the battery conditions or workloads change by simply modifying the image patch size and token keeping rate. Therefore, the hardware efficiency of our SuperViT is very advantageous over the traditional scenario that has to train numerous ViT models in advance, and dynamically download an appropriate model and offload existing one.

4 Experiments

4.1 Implementation Details

We evaluate our SuperViT and compare against state-of-the-art on ImageNet [12]. Following existing studies on ViT compression [31, 43, 27], we use DeiT-S (w/o distillation) [35] and LV-ViT-S [23] as the backbones. All the training strategies, such as data augmentation, regularization and optimizer, strictly follow the original settings of DeiT [35] and LV-ViT [23]. The sequence length in our multi-size splitting includes {8×8, 10×10, 12×12, 14×14} and the multi-token keeping rate includes {1.0, 0.7, 0.5}. We remove less informative tokens at the 4-th,7-th,10-th blocks for both DeiT-S and LV-ViT-S and train SuperViT on a workstation with 4 NVIDIA A100 GPUs.

4.2 Performance Results

Comparison with Backbones. To show the efficacy of our training paradigm, we first list the our SuperViT under different sequence lengths (patch sizes) and token keeping rates in Table 1 and compare it with its backbones of DeiT-S and LV-ViT-S. For fair comparison, we also compare with DeiT-T and LV-ViT-T since our computational costs are tiny in the cases of very small sequence lengths and keeping rates. Our efficacy is measured from two aspects including the Top-1 accuracy to reflect its effectiveness, and FLOPs consumption and model throughput to reflect its efficiency. The model throughput shows the number of processed images per second on a single A100 GPUs [39, 27]. For an accurate throughput estimation, we repeatedly feed each model with a batch size of 512 for 50 times. Then, the practical throughput is computed as $^{512\times50}/_{\text{total inference time}}$.

From Table 1, we observe significant accuracy increase when maintaining similar FLOPs consumption and throughput with backbones. For example, with FLOPs of 4.6G and throughput of 5013, our SuperViT (14×14, 1.0) increases the accuracy of DeiT-S from 79.8% to 80.6%, leading to 0.8% improvement. Also, our SuperViT (14×14, 1.0) gains additional 0.2% accuracy improvement when using LV-ViT-S as the backbone. Besides, when maintaining similar accuracy, SuperViT merits in its significant FLOPs reduction (throughput increase). For example, with accuracy of 79.9% (SuperViT) and 79.8% (DeiT-S), SuperViT saves 28% FLOPs and obtains 1.37× throughput increase, while SuperViT reduces 35% FLOPs and increases throughput by 1.54× with accuracy of 83.2%

Table 1: Efficacy comparison between our SuperViT and its backbones.

Model	Sequence Length	Keeping rate	Top-1 Acc.↑ (%)	FLOPs↓ (G)	Throughput↑ (img./s)
DeiT-T [35]	14×14	1.0	72.2	1.3	5013
SuperViT	8×8	0.5	73.9(+1.7)	0.7(↓46%)	13548(↑2.70×)
SuperViT	8×8	0.7	75.3(+3.1)	1.0(↓23%)	10669(↑2.13×)
SuperViT	8×8	1.0	75.8(+3.6)	1.4(↑8%)	7727(↑1.54×)
SuperViT	10×10	0.5	77.3(+5.1)	1.2(↓8%)	9657(↑1.93×)
SuperViT	10×10	0.7	78.3(+6.1)	1.5(↑15%)	7567(↑1.51×)
SuperViT	10×10	1.0	78.5(+6.2)	2.3(↑77%)	5173(↑1.03×)
SuperViT	12×12	0.5	78.9(+6.7)	1.7(↑31%)	6308(↑1.26×)
DeiT-S [35]	14×14	1.0	79.8	4.6	2461
SuperViT	12×12	0.7	79.6(−0.2)	2.2(↓52%)	4996(↑2.03×)
SuperViT	12×12	1.0	79.9(+0.1)	3.3(↓28%)	3371(↑1.37×)
SuperViT	14×14	0.5	80.0(+0.2)	2.3(↓50%)	4767(↑1.94×)
SuperViT	14×14	0.7	80.5(+0.7)	3.0(↓35%)	3654(↑1.48×)
SuperViT	14×14	1.0	80.6(+0.8)	4.6(↓0%)	2461(↑1.00×)
LV-ViT-T [23]	14×14	1.0	79.1	2.9	3178
SuperViT	8×8	0.5	76.6(−2.5)	1.1(↓62%)	9968(↑3.14×)
SuperViT	8×8	0.7	79.8(+0.7)	1.4(↓52%)	7836(↑2.47×)
SuperViT	8×8	1.0	80.7(+1.6)	2.0(↓31%)	5487(↑1.72×)
SuperViT	10×10	0.5	79.8(+0.7)	1.7(↓41%)	6792(↑2.13×)
SuperViT	10×10	0.7	81.7(+2.6)	2.2(↓24%)	5302(↑1.67×)
SuperViT	10×10	1.0	82.2(+3.1)	3.3(↑14%)	3615(↑1.14×)
SuperViT	12×12	0.5	81.1(+2.0)	2.5(↓14%)	4524(↑1.42×)
SuperViT	14×14	0.5	82.1(+3.0)	3.4(↑17%)	3368(↑1.06×)
LV-ViT-S [23]	14×14	1.0	83.3	6.6	1748
SuperViT	12×12	0.7	82.6(−0.7)	3.2(↓52%)	3565(↑2.04×)
SuperViT	12×12	1.0	82.9(−0.4)	4.7(↓29%)	2357(↑1.35×)
SuperViT	14×14	0.7	83.2(−0.1)	4.3(↓35%)	2684(↑1.54×)
SuperViT	14×14	1.0	83.5(+0.2)	6.6(↓0%)	1748(↑1.00×)

(SuperViT) and 83.3% (LV-ViT-S). Comparing to the tiny version of DeiT and LV-ViT, our SuperViT not only reduces FLOPs and increases throughput, but also significantly enhances the performance.

Comparison with Compressed Models. We continue to compare our efficacy with many studies on model compression including early-existing compression [20, 44, 39] and token drop compression [30, 46, 5, 27, 43, 31, 34]. Early-existing compression methods dynamically choose different inference paths based on a particular criterion. Our SuperViT can also implement early exist since it adapts to multiple complexities. In this part, we give a simple scenario where the cheapest computation (8×8, 0.5) is always used to predict an incoming image and a second one (14×14, 0.7) is utilized again if confidence of the first prediction is smaller than a threshold. By adjusting the threshold, our SuperViT achieves different accuracy-FLOPs trade-offs. Figure 2 plots the performance of our SuperViT built upon DeiT-S and methods including CNN-based MSDNet [20] and RANet [44], as well as transformer-based DVT [39]. Our SuperViT consistently performs better than the state-of-the-art competitor DVT [39]. With similar accuracy, SuperViT results in significantly smaller FLOPs. This is attributed to that the cheapest version of SuperViT already reaches good accuracy of 73.9% in Table 1. Thus most images can be well recognized at very small costs.

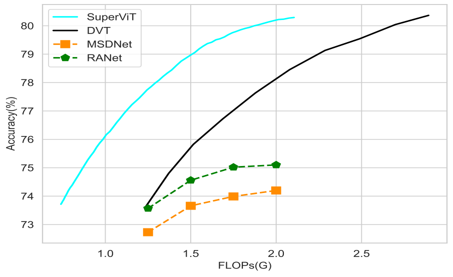


Figure 2: Comparison between SuperViT and early-exiting compression [20, 44, 39].

We go on the comparison with token drop compression, which reduces the number of tokens during network forwarding. For fair comparison, we pick up the results of our SuperViT from Table 1 and compare with the recent state-of-the-arts [30, 46, 5, 27, 43, 31, 34]. Results in Table 2 manifest that

Table 2: Efficacy comparison between our SuperViT and studies on token drops.

Methods	Pre-trained	Top-1 Acc.(%)	FLOPs(G)
DeiT-S (baseline) [35]	-	79.8	4.6
SuperViT(Ours)	✗	**80.6**	**4.6**
IA-RED^2(NeurIPS'2021) [30]	✓	79.1	3.2
A-ViT(CVPR'2022) [46]	✓	78.6	3.6
ViT-Slim(CVPR'2022) [5]	✓	79.9	3.1
Evo-ViT(AAAI'2022) [43]	✗	79.4	3.0
EViT(ICLR'2022) [27]	✗	79.5	3.0
SuperViT(Ours)	✗	**80.5**	**3.0**
DynamicViT (NeurIPS'2021) [31]	✓	79.3	2.9
PS-ViT(CVPR'2022) [34]	✓	79.4	2.6
SuperViT(Ours)	✗	**80.0**	**2.3**
LV-ViT-S (baseline) [23]	-	83.3	6.6
SuperViT(Ours)	✗	**83.5**	**6.6**
DynamicViT(NeurIPS'2021) [31]	✓	83.0	4.6
EViT(ICLR'2022) [27]	✗	83.0	4.7
SuperViT(Ours)	✗	**83.2**	**4.3**

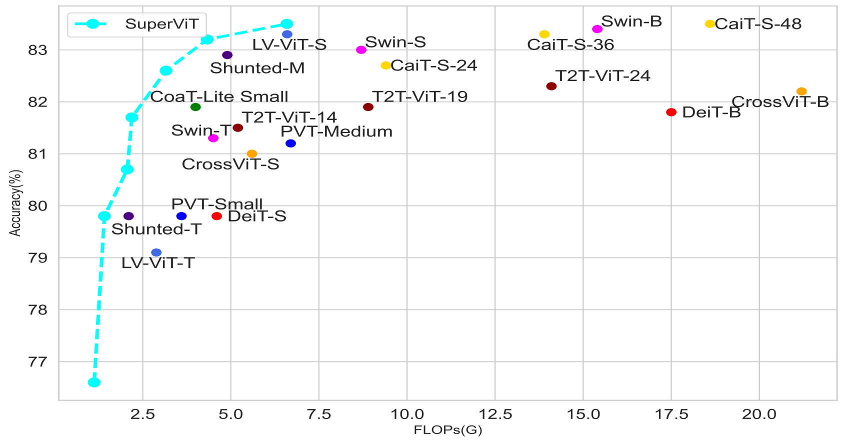
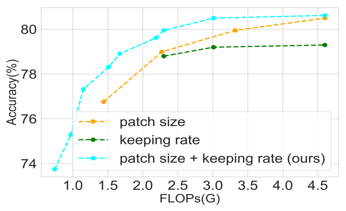


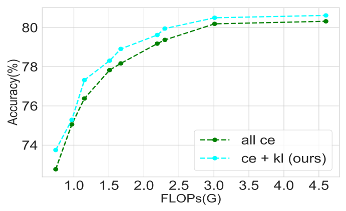
Figure 3: Accuracy and FLOPs trade-off comparison between our SuperViT and popular ViT models.

our SuperViT well outperforms previous methods in both accuracy performance and FLOPs reduction by margins when DeiT-S and LV-ViT-S are used as backbones. For example, upon DeiT-S, our SuperViT significantly outperforms the recent state-of-the-art EViT [27] by 1.1% when consuming the same FLOPs of 3.0G. It is also worth stressing that, existing methods, some of which even heavily rely on a pre-trained model, deteriorate the baseline performance when performing token drops while our SuperViT leads to performance increase in most cases. For example, we reduce 2× FLOPs of DeiT-S while increasing the Top-1 accuracy by 0.2% and 0.7% for 1.5× reduction. These results well demonstrate the effectiveness of our training paradigm to obtain only one ViT model empowered with the ability to recognize images at different levels of computational costs.

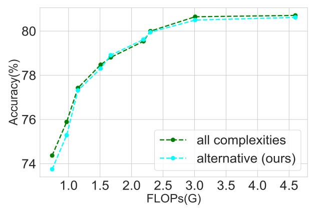
Comparison with ViT Models. In Figure 3, we compare the accuracy and FLOPs trade-off between our SuperViT built upon LV-ViT-S and various ViT models including DeiT [35], PVT [38], CoaT [42], CrossViT [6], Swin [28], T2T-ViT [48], CaiT [36] and Shunted-ViT [32]. The accuracy-FLOPs trade-off of our SuperViT consists of models with different sequence lengths and keeping rates in Table 1. Results in Figure 3 indicate that our SuperViT provides a better accuracy-FLOPs trade-off than majorities of existing ViT models. In fact, these ViTs focus on improving the structure of vanilla ViT or token interactions for accuracy improvement while our SuperViT introduces one new training paradigm. Thus, our SuperViT is orthogonal to these ViT variants, which increases the possibility of integrating our training paradigm into these ViT models. For example, when constructing our SuperViT upon LV-ViT-S, a better accuracy-FLOPs trade-off is obtained in Figure 3.



(a)



(b)



(c)

Figure 4: Ablation studies.

4.3 Ablation Study

In Figure 4a, we first analyze the effectiveness of multi-size patch splitting and multi-token keeping rate respectively. As can be seen, the combination of multi-size patch splitting and multi-token keeping rate results in consistent performance increase compared to these only considering one of them. Our training objective in Eq. (10) adopts cross-entropy loss for the forwarding that does not perform token drops drops drops drops, while Kullback-Leibler divergence is used for the cases where token drops are performed. Figure 4b compares our training objective with the situation where the cross-entropy is considered for all cases. Results show that our training objective is of more benefit to SuperViT. Figure 4c compares the performance of considering all $G \cdot M$ complexities and the alternative of using only four cases (see Sec.3.3). Results show that the former manifests slightly better performance. However, the former takes about 8.8 (DeiT-S backbone) and 14.4 (LV-ViT-S backbone) days to train SuperViT on four NVIDIA A100, while they are 4.1 and 6.5 days for the alternative. Thus, the alternative is more encouraged which is also our implementation standard.[3]

5 Limitation and Future

We further discuss unexplored limitations, which will be our future focus. First, following most compared methods, we verify our SuperViT on the classification task while its efficacy on dense predictions such as detection and segmentation remains unexplored. Second, we construct SuperViT from two dimensions of image patch sizes and token keeping rates. More efforts can be made to take into account the transformer's depth, width of token embedding and so on. Lastly, the vanilla DeiT and LV-ViT consist of plain structures where the number of tokens maintain unchanged in their inputs and outputs. More validations are expected to perform on the pyramid structures [38, 28].

6 Conclusion

Here, we have presented a novel training paradigm to reduce the computational costs in vision transformers (ViTs). We first break down input images into multiple token sequences of different lengths first. In each training iteration, we sequentially feed each token sequence to the network where multiple token keeping rates are imposed as well to learn category prediction distributions at different complexities. Consequently, the trained ViT model, referred to as super vision transformer (SuperViT) in this paper, is demonstrated to provide a better image recognition at a computationally more economical manner compared with existing state-of-the-art methods. Only one SuperViT model is able to process images at different costs thus it also allows efficient hardware utilization in comparison with traditional methods that has to train numerous ViT models in advance.

Acknowledgement

This work was supported by the National Science Fund for Distinguished Young Scholars (No. 62025603), the National Natural Science Foundation of China (No. U21B2037, No. 62176222, No. 62176223, No. 62176226, No. 62072386, No. 62072387, No. 62072389, and No. 62002305), Guangdong Basic and Applied Basic Research Foundation (No. 2019B1515120049), and the Natural Science Foundation of Fujian Province of China (No. 2021J01002).

[3]See appendix for more experiments.

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Int. Conf. Comput. Vis.*, 2021.

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Int. Conf. Mach. Learn.*, 2021.

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *Int. Conf. Learn. Represent.*, 2019.

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Eur. Conf. Comput. Vis.*, 2020.

[5] Arnav Chavan, Zhiqiang Shen, Zhuang Liu, Zechun Liu, Kwang-Ting Cheng, and Eric Xing. Vision transformer slimming: Multi-dimension searching in continuous optimization space. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[6] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[7] Mengzhao Chen, Mingbao Lin, Ke Li, Yunhang Shen, Yongjian Wu, Fei Chao, and Rongrong Ji. Coarse-to-fine vision transformer. *arXiv preprint arXiv:2203.03821*, 2022.

[8] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[9] 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. In *Adv. Neural Inform. Process. Syst.*, 2021.

[10] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021.

[11] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Int. Conf. Comput. Vis.*, pages 248–255, 2009.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. Learn. Represent.*, 2020.

[14] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Int. Conf. Comput. Vis.*, 2021.

[15] 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.

[16] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.

[17] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *Adv. Neural Inform. Process. Syst.*, 2021.

[18] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[19] 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.

[20] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. In *Int. Conf. Learn. Represent.*, 2018.

[21] Lin Huang, Jianchao Tan, Ji Liu, and Junsong Yuan. Hand-transformer: non-autoregressive structured modeling for 3d hand pose estimation. In *Eur. Conf. Comput. Vis.*, 2020.

[22] 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.

[23] Zi-Hang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. In *Adv. Neural Inform. Process. Syst.*, 2021.

[24] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys*, 2021.

[25] Wei Li, Xing Wang, Xin Xia, Jie Wu, Xuefeng Xiao, Min Zheng, and Shiping Wen. Sepvit: Separable vision transformer. *arXiv preprint arXiv:2203.15380*, 2022.

[26] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Int. Conf. Comput. Vis.*, 2021.

[27] Youwei Liang, Chongjian GE, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *Int. Conf. Learn. Represent.*, 2022.

[28] 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 *Int. Conf. Comput. Vis.*, 2021.

[29] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Int. Conf. Comput. Vis.*, 2017.

[30] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. In *Adv. Neural Inform. Process. Syst.*, 2021.

[31] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Adv. Neural Inform. Process. Syst.*, 2021.

[32] Sucheng Ren, Daquan Zhou, Shengfeng He, Jiashi Feng, and Xinchao Wang. Shunted self-attention via multi-scale token aggregation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[33] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Int. Conf. Comput. Vis.*, 2017.

[34] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[35] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Int. Conf. Mach. Learn.*, 2021.

[36] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Int. Conf. Comput. Vis.*, 2021.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst.*, 2017.

[38] 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 *Int. Conf. Comput. Vis.*, 2021.

[39] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In *Adv. Neural Inform. Process. Syst.*, 2021.

[40] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. *arXiv preprint arXiv:2201.00520*, 2022.

[41] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Adv. Neural Inform. Process. Syst.*, 2021.

[42] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[43] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In *AAAI Conf. Artificial Intelli.*, 2022.

[44] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2369–2378, 2020.

[45] Taojiannan Yang, Sijie Zhu, Chen Chen, Shen Yan, Mi Zhang, and Andrew Willis. Mutualnet: Adaptive convnet via mutual learning from network width and resolution. In *Eur. Conf. Comput. Vis.*, pages 299–315, 2020.

[46] Hongxu Yin, Arash Vahdat, Jose Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[47] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *Int. Conf. Learn. Represent.*, 2018.

[48] 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 *Int. Conf. Comput. Vis.*, 2021.

[49] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[50] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[51] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[52] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *Int. Conf. Learn. Represent.*, 2022.

[53] Yichen Zhu, Yuqin Zhu, Jie Du, Yi Wang, Zhicai Ou, Feifei Feng, and Jian Tang. Make a long image short: Adaptive token length for vision transformers. *arXiv preprint arXiv:2112.01686*, 2021.

Appendix

Comparison with Individual ViT Training. In this part, we further provide additional experiments to show the efficacy of our training paradigm over traditional individual ViT training. As stated in Sec. 4.1 of the main paper, the sequence length in our multi-size splitting includes {8×8, 10×10, 12×12, 14×14} and the multi-token keeping rate includes {1.0, 0.7, 0.5}, resulting in a total of 12 different computational costs. Our training paradigm sequentially forwards one of these 12 cases, which leads to only one SuperViT model that provides improved image recognition performance with various computational costs. In contrast, traditional methods have to train 12 individual ViT models of different complexities.

Compared with traditional methods, our SuperViT merits in three aspects. First, significant performance improvement is observed. Table 3 shows the accuracy comparison between our SuperViT and its counterpart that trains each ViT model individually. As can be seen, under the same sequence

Table 3: Efficacy comparison between our training paradigm and individual training with DeiT-S [35] as the backbone. The † denotes models are trained individually.

Sequence Length	Keeping rate	DeiT-S† Top-1 Acc.	SuperViT Top-1 Acc.(↑)
8×8	0.5	71.6%	73.9%(2.3%↑)
8×8	0.7	73.5%	75.3%(1.8%↑)
8×8	1.0	74.8%	75.8%(1.0%↑)
10×10	0.5	74.8%	77.3%(2.5%↑)
10×10	0.7	76.7%	78.3%(1.6%↑)
10×10	1.0	77.4%	78.5%(1.1%↑)
12×12	0.5	76.5%	78.9%(2.4%↑)
12×12	0.7	78.0%	79.6%(1.6%↑)
12×12	1.0	78.7%	79.9%(1.2%↑)
14×14	0.5	78.0%	80.0%(2.0%↑)
14×14	0.7	79.2%	80.5%(1.3%↑)
14×14	1.0	79.8%	80.6%(0.8%↑)

length and keeping rate, our SuperViT increases the performance by 0.8% $\sim$ 2.5%. In particular, over 2.0% performance gains are obtained under a small keeping rate of 0.5. Second, our training paradigm is more economical in training consumption. It takes around 4.1 days to train our SuperViT on a workstation with 4 NVIDIA A100 GPUs when using DeiT-S [35] as the backbone while about 24.4 days are required for traditional individual training. Third, our SuperViT is of more hardware efficiency. As discussed in Sec. 3.3 of the main paper, SuperViT can be executable on various hardware platforms equipped with different resources since it can recognize images at different computational costs. Also, SuperViT allows instant and adaptive accuracy-efficiency trade-offs by simply adjusting the image patch size and token keeping rate once the available resource on the same platform changes while traditional methods have to download an appropriate model and offload existing one.