Not All Images are Worth 16x16 Words: Dynamic Vision Transformers with Adaptive Sequence Length

Yulin Wang[1*] **Rui Huang**[1*] **Shiji Song**[1] **Zeyi Huang**[2] **Gao Huang**[1†]
[1]Department of Automation, BNRist, Tsinghua University, Beijing, China
[2]Huawei Technologies Ltd., China
{wang-yl19, hr20}@mails.tsinghua.edu.cn, huangzeyi2@huawei.com.
{shijis, gaohuang}@tsinghua.edu.cn

Abstract

Vision Transformers (ViT) have achieved remarkable success in large-scale image recognition. They split every 2D image into a fixed number of patches, each of which is treated as a token. Generally, representing an image with more tokens would lead to higher prediction accuracy, while it also results in drastically increased computational cost. To achieve a decent trade-off between accuracy and speed, the number of tokens is empirically set to 16x16. In this paper, we argue that every image has its own characteristics, and ideally the token number should be conditioned on each individual input. In fact, we have observed that there exist a considerable number of "easy" images which can be accurately predicted with a mere number of 4x4 tokens, while only a small fraction of "hard" ones need a finer representation. Inspired by this phenomenon, we propose a Dynamic Transformer to automatically configure a proper number of tokens for each input image. This is achieved by cascading multiple Transformers with increasing numbers of tokens, which are sequentially activated in an adaptive fashion at test time, i.e., the inference is terminated once a sufficiently confident prediction is produced. We further design efficient feature reuse and relationship reuse mechanisms across different components of the Dynamic Transformer to reduce redundant computations. Extensive empirical results on ImageNet, CIFAR-10, and CIFAR-100 demonstrate that our method significantly outperforms the competitive baselines in terms of both theoretical computational efficiency and practical inference speed.

1 Introduction

Transformers, the dominant self-attention-based models in natural language processing (NLP) [10, 37, 3], have been successfully adapted to image recognition problems [11, 52, 35, 16] recently. In particular, vision Transformers achieve state-of-the-art performance on the large scale ImageNet benchmark [9], while exhibit excellent scalability with the further growing dataset size (e.g., on JFT-300M [11]). These models split each image into a fixed number of patches and embed them into 1D tokens as inputs. Typically, representing the data using more tokens contributes to higher prediction accuracy, but leads to intensive computational cost, which grows quadratically with respect to the token number in self-attention blocks. For a proper trade-off between efficiency and effectiveness, existing works empirically adopt 14x14/16x16 tokens [11, 52].

In this paper, we argue that it may not be optimal to treat all samples with the same number of tokens. In fact, there exist considerable variations among different images (e.g., contents, scales of objects, backgrounds, etc.). Therefore, the number of representative tokens should ideally be configured specifically for each input. This issue is critical for the computational

Table 1: Accuracy and computational cost of T2T-ViT-12 with different token numbers on ImageNet.

# of Tokens	14x14	7x7	4x4
Accuracy	76.7%	70.3%	60.8%
FLOPs	1.78G	0.47G	0.21G

[*]Equal contribution.
[†]Corresponding author.

efficiency of the models. For example, we train a T2T-ViT-12 [52] with varying token numbers, and report the corresponding accuracy and FLOPs in Table 1. One can observe that adopting the officially recommended 14x14 tokens only correctly recognizes $\sim$15.9% (76.7% v.s. 60.8%) more test samples compared to that of using 4x4 tokens, while increases the computational cost by 8.5 times (1.78G v.s. 0.21G). In other words, computational resources are wasted on applying the unnecessary 14x14 tokens to many "easy" images for which 4x4 tokens are sufficient.

Motivated by this observation, we propose a novel *Dynamic Vision Transformer* (DVT) framework, aiming to automatically configure a decent token number conditioned on each image for high computational efficiency. In specific, a cascade of Transformers are trained using increasing number of tokens. At test time, these models are sequentially activated starting with less tokens. Once a prediction with sufficient confidence has been produced, the inference procedure will be terminated immediately. As a consequence, the computation is unevenly allocated among "easy" and "hard" samples by adjusting the token number, yielding a considerable improvement in efficiency. Importantly, we further develop *feature-wise* and *relationship-wise* reuse mechanisms to reduce redundant computations. The former allows the downstream models to be trained on the basis of previously extracted deep features, while the later enables leveraging existing upstream self-attention relationships to learn more accurate attention maps. Illustrative examples of our method are given in Figure 1.

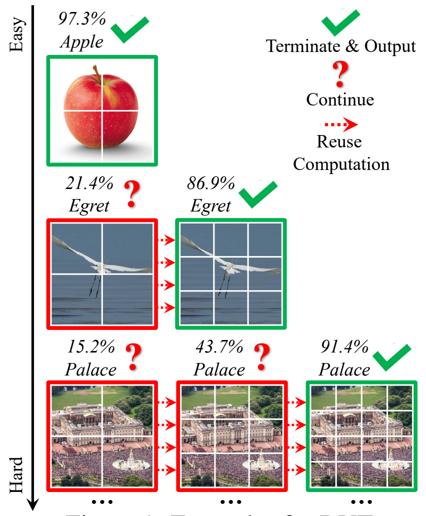

Figure 1: Examples for DVT.

Notably, DVT is designed as a general framework. Most of the state-of-the-art image recognition Transformers, such as ViT [11], DeiT [35], and T2T-ViT [52], can be straightforwardly deployed as its backbones for higher efficiency. Our method is also appealing in its flexibility. The computational cost of DVT is able to be adjusted online by simply adapting the early-termination criterion. This characteristic makes DVT suitable for the cases where the available computational resources fluctuate dynamically or a minimal power consumption is required to achieve a given performance. Both situations are ubiquitous in real-world applications (e.g., searching engines and mobile apps).

The performance of DVT is evaluated on ImageNet [9] and CIFAR [25] with T2T-ViT [52] and DeiT [35]. Experimental results show that DVT significantly improves the efficiency of the backbones. For examples, DVT reduces the computational cost of T2T-ViT by 1.6-3.6x without sacrificing accuracy. The real inference speed on a NVIDIA 2080Ti GPU is consistent with our theoretical results.

2 Related Work

Vision Transformers. Inspired by the success of Transformers on NLP tasks [10, 37, 3, 40], vision Transformers (ViT) have recently been developed for image recognition [11]. Although ViT by itself is not comparable with state-of-the-art convolutional networks (CNN) on the standard ImageNet benchmark, it attains excellent results when pre-trained on the larger JFT-300M dataset. DeiT [35] studies the training strategy of ViT and proposes a knowledge distilling-based approach, surpassing the performance of ResNet [18]. Some following works such as T2T-ViT [52], TNT [16], CaiT [36], DeepViT [59], CPVT [6], LocalViT [27] and CrossViT [5] focus on improving the architecture design of ViT. Another line of research proposes to integrate the inductive bias of CNN into Transformers [46, 8, 51, 15]. There are also attempts to adapt ViT for other vision tasks (e.g., object detection, semantic segmentation, etc.) [29, 41, 12, 19, 54, 57]. The most majority of these concurrent works represent each image with a fix number of tokens. To the best of our knowledge, we are the first to consider configuring token numbers conditioned on the inputs.

Efficient deep networks. Computational efficiency plays a critical role in real-world scenarios, where the executed computation translates into power consumption, carbon emission or latency. A number of works have been done on reducing the computational cost of CNNs [21, 31, 20, 50, 56, 30, 33]. However, designing efficient vision Transformers is still an under-explored topic. T2T-ViT [52] proposes a light-weighted tokens-to-token module and obtains a competitive accuracy-parameter trade-off compared to MobileNetV2 [31]. LeViT [15] accelerates the inference of Transformer models by involving convolutional layers. Swin Transformer [29] introduces an efficient shifted

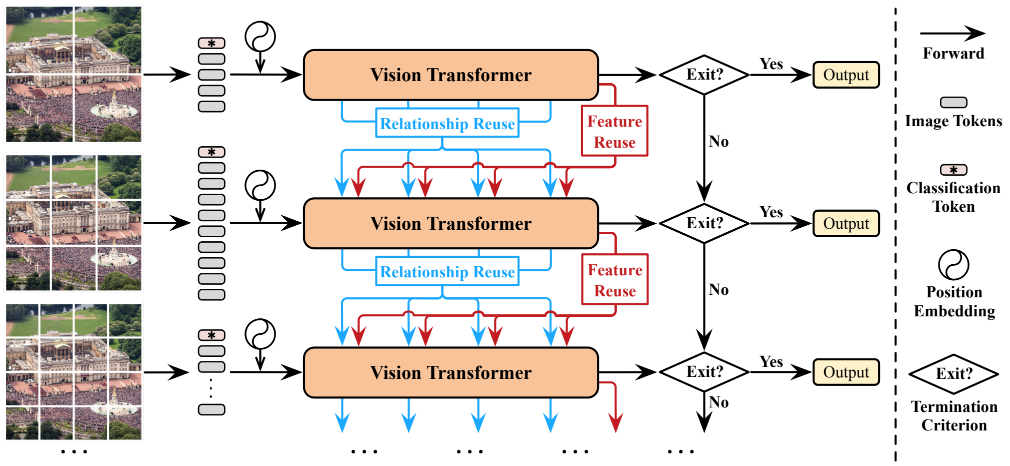

Figure 2: An overview of *Dynamic Vision Transformers* (DVT). Under the objective of configuring proper token numbers conditioned on the inputs, we cascade multiple Transformers with increasing number of tokens. At test time, they are sequentially activated until a convincing prediction (e.g. sufficiently confident) has been obtained or the final model has been inferred. The feature and relationship reuse mechanisms allow reusing computation across different Transformers.

window-based approach in multi-stage vision Transformers. Compared to these models with fixed computational graphs, the proposed DVT framework improves the efficiency by adaptively changing the architecture of the network on a per-sample basis.

Dynamic models. Designing dynamic architectures is an effective approach for efficient deep learning [17]. In the context of recognition tasks, MSDNet and its variants [22, 49, 26] develop a multi-classifier CNN architecture to perform early exiting for easy samples. Another type of dynamic CNNs skips redundant layers [38, 42, 48] or channels [28] conditioned on the inputs. Besides, the spatial adaptive paradigm [14, 4, 44, 39, 43] has been proposed for efficient image and video recognition. Although these works are related to DVT on the spirit of adaptive computation, they are developed based on CNN, while DVT is tailored for vision Transformers.

3 Dynamic Vision Transformer

Vision Transformers [11, 16, 35, 52] split each 2D image into 1D tokens, while model their long range interaction with the self-attention mechanism [37]. As aforementioned, to correctly recognize some "hard" images and achieve high accuracy, the number of tokens usually needs to be large, leading to the quadratically grown computational cost. However, "easier" images that make up the bulk of the datasets typically require far fewer tokens and much less costs (as shown in Table 1). Inspired by this observation, we propose a *Dynamic Vision Transformers* (DVT), aiming to improve the computational efficiency of Transformers via adaptively reducing the number of representative tokens for each input.

In specific, we propose to deploy multiple Transformers trained with increasing number of tokens, such that one can sequentially activate them for each test image until obtaining a convincing prediction (e.g., with sufficient confidence). The computation is allocated unevenly across different samples for improving the overall efficiency. It is worth noting that, if all the Transformers are learned separately, the computation performed by upstream models will simply be abandoned once a downstream Transformer is activated, resulting in considerable inefficiency. To alleviate this problem, we introduce the efficient feature and relationship reuse mechanisms.

3.1 Overview

Inference. We start by describing the inference procedure of DVT, which is shown in Figure 2. For each test sample, we first coarsely represent it using a small number of 1D token embeddings. This can be achieved by either straightforwardly flattening the split image patches [11, 16] or leveraging techniques like the tokens-to-token module [52]. We infer a vision Transformer with these few tokens to obtain a quick prediction. This process enjoys high efficiency since the computational cost of Transformers grows quadratically with respect to token number. Then the prediction will be evaluated with certain criterion to determine whether it is reliable enough to be retrieved immediately. In this paper, early-termination is performed when the model is sufficiently confident (details in Section 3.3).

Once the prediction fails to meet the termination criterion, the original input image will be split into more tokens for more accurate but computationally more expensive inference. Note that, here the dimension of each token embedding remains unchanged, while the number of tokens increases, enabling more fine-grained representation. An additional Transformer with the same architecture as the previous one but different parameters will be activated. By design, this stage trades off computation for higher accuracy on some "difficult" test samples. To improve the efficiency, the new model can reuse the previously learned features and relationships, which will be introduced in Section 3.2. Similarly, after obtaining a new prediction, the termination criterion will be applied, and the above procedure will proceed until the sample exits or the final Transformer has been inferred.

Training. For training, we simply train DVT to produce correct predictions at all exits (i.e., each with the corresponding number of tokens). Formally, the optimization objective is

$$\text{minimize} \ \frac{1}{|\mathcal{D}_{\text{train}}|} \sum\nolimits_{(\boldsymbol{x}, y) \in \mathcal{D}_{\text{train}}} \left[\sum\nolimits_{i} L_{\text{CE}}(\boldsymbol{p}_i, y) \right], \tag{1}$$

where $(\boldsymbol{x}, y)$ denote a sample in the training set $\mathcal{D}_{\text{train}}$ and its corresponding label. We adopt the standard cross-entropy loss function $L_{\text{CE}}(\cdot)$, while $\boldsymbol{p}_i$ denotes the softmax prediction probability output by the i^{th} exit. We find that such a simple training objective works well in practice.

Transformer backbone. DVT is proposed as a general and flexible framework. It can be built on top of most existing vision Transformers like ViT [11], DeiT [35] and T2T-ViT [52] to improve their efficiency. The architecture of Transformers simply follows the implementation of these backbones.

3.2 Feature and Relationship Reuse

An important challenge to develop our DVT approach is how to facilitate the *reuse* of computation. That is, once a downstream Transformer with more tokens is inferred, it is obviously inefficient if the computation performed in previous models is abandoned. The upstream models, although being based on smaller number of input tokens, are trained with the same objective, and have extracted valuable information for fulfilling the task. Therefore, we propose two mechanisms to reuse the learned deep features and self-attention relationships. Both of them are able to improve the test accuracy significantly by involving minimal extra computational cost.

Background. For the ease of introduction, we first revisit the basic formulation of vision Transformers. The Transformer encoders consist of alternatively stacked multi-head self-attention (MSA) and multi-layer perceptron (MLP) blocks [37, 11]. The layer normalization (LN) [2] and residual connection [18] are applied before and after each block, respectively. Let $\boldsymbol{z}_l \in \mathbb{R}^{N \times D}$ denote the output of the l^{th} Transformer layer, where N is the number of tokens for each sample, and D is the dimension of each token. Note that $N = HW + 1$, which corresponds to $H \times W$ patches of the original image and a single learnable classification token. Formally, we have

$$\boldsymbol{z}'_l = \text{MSA}(\text{LN}(\boldsymbol{z}_{l-1})) + \boldsymbol{z}_{l-1}, \qquad l \in \{1, \ldots, L\}, \tag{2}$$

$$\boldsymbol{z}_l = \text{MLP}(\text{LN}(\boldsymbol{z}'_l)) + \boldsymbol{z}'_l, \qquad l \in \{1, \ldots, L\}, \tag{3}$$

where L is the total number of layers in the Transformer. The classification token in $\boldsymbol{z}_L$ will be fed into a LN layer followed by a fully-connected layer for the final prediction. For simplicity, here we omit the details on the position embedding, which is unrelated to our main idea. No modification is performed on it in addition to the configurations of backbones.

Feature reuse. All the Transformers in DVT share the same goal of extracting discriminative representations for accurate recognition. Therefore, it is straightforward that downstream models should be learned on the basis of previously obtained deep features, rather than extracting features from scratch. The former is more efficient since the computation performed in an upstream model contributes to both itself and the successive models. To implement this idea, we propose a feature reuse mechanism (see: Figure 3). In specific, we leverage the image tokens output by the final layer of the upstream Transformer, i.e., $\boldsymbol{z}_L^{\text{up}}$, to learn a layer-wise embedding $\mathbf{E}_l$ for the downstream model:

$$\mathbf{E}_l = f_l(\boldsymbol{z}_L^{\text{up}}) \in \mathbb{R}^{N \times D'}. \tag{4}$$

Herein, f_l: $\mathbb{R}^{N \times D} \to \mathbb{R}^{N \times D'}$ consists of a sequence of operations starting with a LN-MLP ($\mathbb{R}^D \to \mathbb{R}^{D'}$), which introduces nonlinearity and allows more flexible transformations. Then the image tokens are

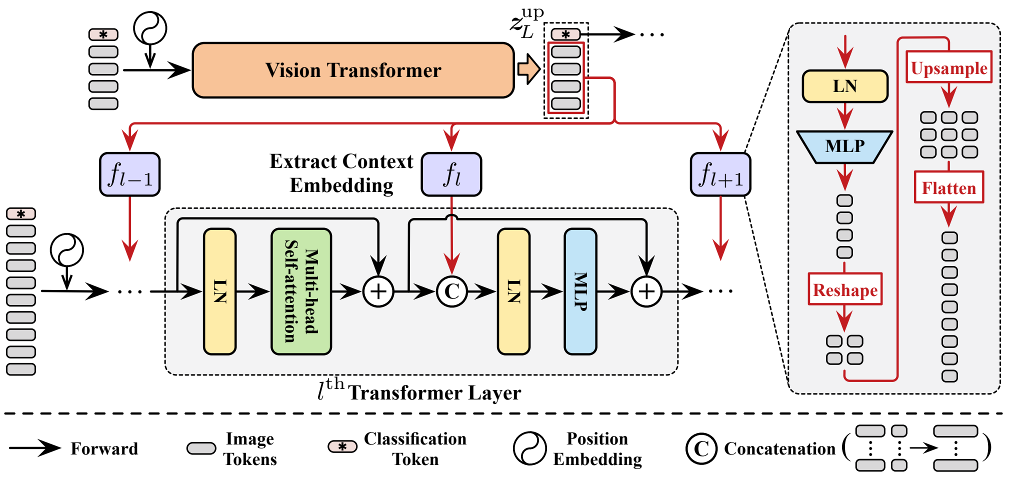

Figure 3: Illustration of the feature reuse mechanism. A layer-wise context embedding is learned based on the final representations output by the upstream model, i.e., z_L^{up}, and integrated into the MLP block of each downstream Transformer layer.

reshaped to the corresponding locations in the original image, upsampled and flattened to match the token number of the downstream model. Typically, we use a small D' for an efficient f_l.

Consequently, the embedding $\mathbf{E}_l$ is injected into the downstream model, providing prior knowledge on recognizing the input image. Formally, we replace Eq. (3) by:

$$z_l = \text{MLP}(\text{LN}(\text{Concat}(z'_l, \mathbf{E}_l))) + z'_l, \qquad l \in \{1, \dots, L\}, \tag{5}$$

where $\mathbf{E}_l$ is concatenated with the intermediate tokens z'_l. We simply increase the dimension of LN and the first layer of MLP from D to $D+D'$. Since $\mathbf{E}_l$ is based on the upstream outputs z_L^{up} that have less tokens than z'_l, it actually concludes the context information of the input image for each token in z'_l. Therefore, we name $\mathbf{E}_l$ as the *context embedding*. Besides, we do not reuse the classification token and pad zero for it in Eq. (5), which we empirically find beneficial for the performance. Intuitively, Eqs. (4) and (5) allow training the downstream model to flexibly exploit the information within z_L^{up} on a per-layer basis, under the objective of minimizing the final recognition loss (Eq. (1)). This feature reuse formulation can also be interpreted as implicitly enlarging the depth of the model.

Relationship reuse. A prominent advantage of vision Transformers is that their self-attention blocks enable integrating information across the entire image, which effectively models the long-range dependencies in the data. Typically, the models need to learn a group of attention maps at each layer to describe the relationships among tokens. Apart from the deep features mentioned above, the downstream models also have access to the self-attention maps produced in previous models. We argue that these learned relationships are also capable of being reused to facilitate the learning of downstream Transformers.

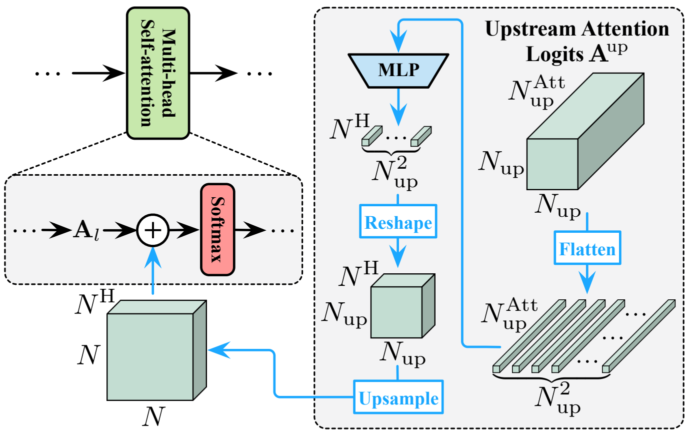

Figure 4: Illustration of the relationship reuse mechanism. We leverage the learned self-attention relationships from all upstream layers and attention heads, i.e., $\mathbf{A}^{\text{up}}$, to refine the downstream attention maps. The addition operation of logits is adopted. Note that N^{H} denotes the head number for multi-head self-attention.

Given the input representation z_l, the self-attention is performed as follows. First, the query, key and value matrices $\mathbf{Q}_l$, $\mathbf{K}_l$ and $\mathbf{V}_l$ are computed via linear projections:

$$\mathbf{Q}_l = z_l \mathbf{W}_l^{\text{Q}}, \quad \mathbf{K}_l = z_l \mathbf{W}_l^{\text{K}}, \quad \mathbf{V}_l = z_l \mathbf{W}_l^{\text{V}}, \tag{6}$$

where $\mathbf{W}_l^{\mathrm{Q}}$, $\mathbf{W}_l^{\mathrm{K}}$ and $\mathbf{W}_l^{\mathrm{V}}$ are weight matrices. Then the attention map is calculated by a scaled dot-product operation with softmax to aggregate the values of all tokens, namely

$$\mathrm{Attention}(\boldsymbol{z}_l) = \mathrm{Softmax}(\mathbf{A}_l)\mathbf{V}_l, \quad \mathbf{A}_l = \mathbf{Q}_l\mathbf{K}_l^{\top}/\sqrt{d}. \tag{7}$$

Here d is the hidden dimension of $\mathbf{Q}$ or $\mathbf{K}$, and $\mathbf{A}_l \in \mathbb{R}^{N\times N}$ denotes the logits of the attention map. Note that we omit the details on the multi-head attention mechanism for clarity, where $\mathbf{A}_l$ may include multiple attention maps. Such a simplification does not affect the description of our method.

For relationship reuse, we first concatenate the attention logits produced by all layers of the upstream model (i.e., $\mathbf{A}_l^{\mathrm{up}}, l \in \{1, \ldots, L\}$):

$$\mathbf{A}^{\mathrm{up}} = \mathrm{Concat}(\mathbf{A}_1^{\mathrm{up}}, \mathbf{A}_2^{\mathrm{up}}, \ldots, \mathbf{A}_L^{\mathrm{up}}) \in \mathbb{R}^{N_{\mathrm{up}}\times N_{\mathrm{up}}\times N_{\mathrm{up}}^{\mathrm{Att}}}, \tag{8}$$

where N_{up} and $N_{\mathrm{up}}^{\mathrm{Att}}$ denote the number of tokens and all attention maps in the upstream model, respectively. Typically, we have $N_{\mathrm{up}}^{\mathrm{Att}} = N^{\mathrm{H}}L$, where N^{H} is the number of heads for the multi-head attention and L is the number of layers. Then the downstream Transformer learns attention maps by leveraging both its own tokens and $\mathbf{A}^{\mathrm{up}}$ simultaneously. Formally, we replace Eq. (7) by

$$\mathrm{Attention}(\boldsymbol{z}_l) = \mathrm{Softmax}(\mathbf{A}_l + r_l(\mathbf{A}^{\mathrm{up}}))\mathbf{V}_l, \quad \mathbf{A}_l = \mathbf{Q}_l\mathbf{K}_l^{\top}/\sqrt{d}, \tag{9}$$

where $r_l(\cdot)$ is a transformation network that integrates the information provided by $\mathbf{A}^{\mathrm{up}}$ to refine the downstream attention logits $\mathbf{A}_l$. The architecture of $r_l(\cdot)$ is presented in Figure 4, which includes a MLP for nonlinearity followed by an upsample operation to match the size of attention maps. For multi-head attention, the output dimension of the MLP will be set to the number of heads.

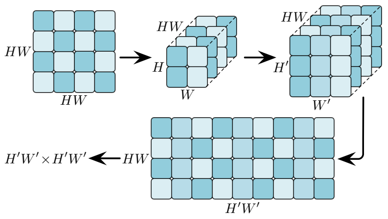

Figure 5: An example for the upsample operation in $r_l(\cdot)$.

Notably, Eq. (9) is a simple but flexible formulation. For one thing, each self-attention block in the downstream model has access to all the upstream attention heads in both shallow and deep layers, and hence can be trained to leverage multi-level relationship information on its own basis. For another, as the newly generated attention maps and the reused relationships are combined in logits, their relative importance can be automatically learned by adjusting the magnitude of logits. It is also worth noting that the regular upsample operation cannot be directly applied in $r_l(\cdot)$. To illustrate this issue, we take upsampling a $HW\times HW$ $(H=W=2)$ attention map to $H'W'\times H'W'$ $(H'=W'=3)$ for example in Figure 5. Since each of its rows and columns corresponds to $H\times W$ image tokens, we reshape the rows or columns back to $H\times W$, scale them to $H'\times W'$, and then flatten them to $H'W'$ vectors.

3.3 Adaptive Inference

As aforementioned, the proposed DVT framework progressively increases the number of tokens for each test sample and performs early-termination, such that "easy" and "hard" images can be processed using varying tokens with uneven computational cost, improving the overall efficiency. Specifically, at the i^{th} exit that produces the softmax prediction $\boldsymbol{p}_i$, the largest entry of $\boldsymbol{p}_i$, i.e., $\max_j p_{ij}$ (defined as confidence [22, 49, 44]), is compared with a threshold η_i. If $\max_j p_{ij} \geq \eta_i$, the inference will stop by adopting $\boldsymbol{p}_i$ as the output. Otherwise, the image will be represented using more tokens to activate the downstream Transformer. We always adopt a zero-threshold for the final Transformer.

The values of $\{\eta_1, \eta_2, \ldots\}$ are solved on the validation set. We assume a *budgeted batch classification* [22] setting, where DVT needs to recognize a set of samples $\mathcal{D}_{\mathrm{val}}$ within a given computational budget $B > 0$. Let $\mathrm{Acc}(\mathcal{D}_{\mathrm{val}}, \{\eta_1, \eta_2, \ldots\})$ and $\mathrm{FLOPs}(\mathcal{D}_{\mathrm{val}}, \{\eta_1, \eta_2, \ldots\})$ denote the accuracy and computational cost on $\mathcal{D}_{\mathrm{val}}$ when using the thresholds $\{\eta_1, \eta_2, \ldots\}$. The optimal thresholds can be obtained by solving the following optimization problem:

$$\underset{\eta_1,\eta_2,\ldots}{\mathrm{maximize}} \ \mathrm{Acc}(\mathcal{D}_{\mathrm{val}}, \{\eta_1, \eta_2, \ldots\}), \qquad s.t. \ \mathrm{FLOPs}(\mathcal{D}_{\mathrm{val}}, \{\eta_1, \eta_2, \ldots\}) \leq B. \tag{10}$$

Due to the non-differentiability, we solve this problem with the genetic algorithm [45] in this paper.

4 Experiments

In this section, we empirically validate the proposed DVT on ImageNet [9] and CIFAR-10/100 [25]. Ablation studies and visualization are presented on ImageNet to give a deeper understanding of our method. Code and pre-trained models will be available at `https://github.com/blackfeather-wang/Dynamic-Vision-Transformer`.

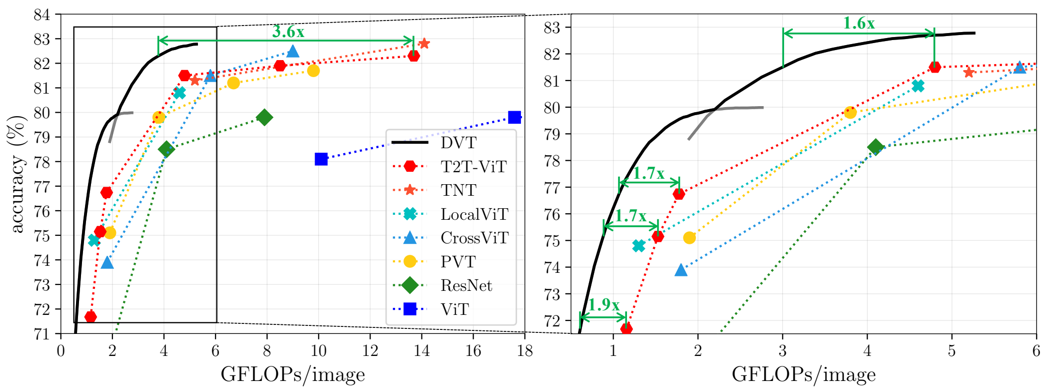

Figure 6: Top-1 accuracy v.s. GFLOPs on ImageNet. DVT is implemented on top of T2T-ViT-12/14.

Table 2: The practical speed of DVT.

Models	ImageNet (NVIDIA 2080Ti, bs=128) Top-1 acc.	Throughput
T2T-ViT-7	71.68%	1574 img/s
DVT	**78.48%** (↑6.80%)	1574 img/s
T2T-ViT-10	75.15%	1286 img/s
DVT	**79.74%** (↑4.59%)	1286 img/s
T2T-ViT-12	76.74%	1121 img/s
DVT	**80.43%** (↑3.69%)	1128 img/s
T2T-ViT-14	81.50%	619 img/s
DVT	81.50%	**877 img/s** (↑1.42x)
T2T-ViT-19	81.93%	382 img/s
DVT	81.93%	**666 img/s** (↑1.74x)

Table 3: Performance of DVT on CIFAR-10/100.

Models	CIFAR-10		CIFAR-100	
	Top-1 acc.	GFLOPs	Top-1 acc.	GFLOPs
T2T-ViT-10	97.21%	1.53	85.44%	1.53
DVT	97.21%	**0.50** (↓3.1x)	85.45%	**0.54** (↓2.8x)
T2T-ViT-12	97.45%	1.78	86.23%	1.78
DVT	97.46%	**0.52** (↓3.4x)	86.26%	**0.61** (↓2.9x)
T2T-ViT-14	98.19%	4.80	89.10%	4.80
DVT	98.19%	**0.77** (↓6.2x)	89.11%	**1.62** (↓3.0x)
T2T-ViT-19	98.43%	8.50	89.37%	8.50
DVT	98.43%	**1.44** (↓5.9x)	89.38%	**1.74** (↓4.9x)
T2T-ViT-24	98.53%	13.69	89.62%	13.69
DVT	98.53%	**1.49** (↓9.2x)	89.63%	**1.86** (↓7.4x)

Datasets. (1) ImageNet is a 1,000-class dataset from ILSVRC2012 [9], containing 1.2 million images for training and 50,000 images for validation. (2) CIFAR-10/100 datasets [25] contain 32x32 colored images in 10/100 classes. Both of them consist of 50,000 images for training and 10,000 images for testing. For all the three datasets, we adopt the same data pre-processing and data augmentation policy as [18, 23, 22]. In addition, we solve the confidence thresholds stated in Section 3.3 on the training set, which we find achieves similar performance to adopting cross-validation.

Backbones. Our experiments are based on several state-of-the-art vision Transformers, namely T2T-ViT-12 [52], T2T-ViT-14 [52], and DeiT-small (w/o distillation) [35]. Unless otherwise specified, we deploy DVT with three exits, corresponding to representing the images as 7x7, 10x10 and 14x14 tokens[3]. For fair comparisons, our implementation exploits the official code of the backbones, and adopts exactly the same training hyper-parameters. More training details can be found in Appendix A. The number of FLOPs is calculated using the *fvcore* toolkit provided by Facebook AI Research, which is also used in Detectron2 [47], PySlowFast [13], and ClassyVision [1].

Implementation details. For feature reuse, the hidden size and output size of the MLP in $f_l(\cdot)$ are set to 128 and 48. In relationship reuse, for implementation efficiency, we share the same hidden state across the MLPs of all $r_l(\cdot)$, such that $r_l(\mathbf{A}^{\mathrm{up}}), l \in \{1, \ldots, L\}$ can be obtained at one time in concatenation by implementing a single large MLP, whose hidden size and output size are $3N^{\mathrm{H}}L$ and $N^{\mathrm{H}}L$. Note that N^{H} is the head number of multi-head attention and L is the layer number.

4.1 Main Results

Results on ImageNet are shown in Figures 6 and 7, where T2T-ViT [52] and DeiT [35] are implemented as backbones respectively. As stated in Section 3.3, we vary the average computational budget, solve the confidence thresholds, and evaluate the corresponding validation accuracy. The performance of DVT is plotted in gray curves, with the best accuracy under each budget plotted in black curves. We also compare our method with several highly competitive baselines, i.e., TNT [16], LocalViT [27], CrossViT [5], PVT [41], ViT [11] and ResNet [18]. It can be observed that DVT consistently reduces the computational cost of the backbones. For example, DVT achieves the 82.3% accuracy with 3.6x less FLOPs compared with the vanilla T2T-ViT. When the budget ranges among

[3]Although 4x4 tokens are also used as an example in Section 1, we find starting with 7x7 is more efficient.

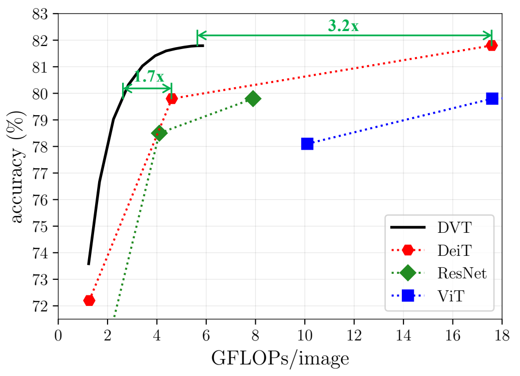

Figure 7: Performance of DeiT-based DVT on ImageNet. DeiT-small is used as the backbone.

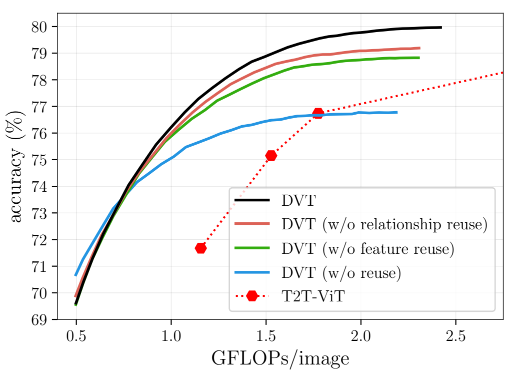

Figure 8: Performance of the DVT based on T2T-ViT-12 with and without the reuse mechanisms.

0.5-2 GFLOPs, DVT has ~1.7-1.9x less computation than T2T-ViT with the same performance. Notably, our method can flexibly attain all the points on each curve by simply adjusting the values of confidence thresholds with a single DVT.

Practical efficiency of DVT. We test the actual speed of DVT on a NVIDIA 2080Ti GPU under a batch inference setting, where a mini-batch of data is fed into the model at a time. After inferring each Transformer, the samples that meet the early-termination criterion will exit, with the remaining images fed into the downstream Transformer. The results are presented in Table 2. Here we adopt a two-exit DVT based on T2T-ViT-12 using 7x7 and 14x14 tokens, which we find more efficient in practice. All other implementation details remain unchanged. One can observe that DVT improves the accuracy of small models (T2T-ViT-7/10/12) by 3.7-6.8% with the same inference speed, while accelerates the inference of the large T2T-ViT-14/19 models by 1.4-1.7x without sacrificing performance.

Results on CIFAR are presented in Table 3. Following the common practice [11, 52, 16, 35], we resize the CIFAR images to 224x224, and fine-tune the T2T-ViT and DVT models in Figure 6. The official code and training configurations provided by [52] are utilized. We report the computational cost of DVT when it achieves the competitive performance with baselines. Our proposed method is shown to consume ~3-9x less computation compared with T2T-ViT.

4.2 Ablation Study

Effectiveness of feature and relationship reuse. We conduct experiments by ablating one or both of the reuse mechanisms. For a clear comparison, we first deactivate the early-termination, and report the accuracy and GFLOPs corresponding to each exit in Table 4. The three-exit DVT based on T2T-ViT-12 is considered. One can observe that both the two reuse mechanisms are able to significantly boost the accuracy of DVT at the 2nd and 3rd exits with at most 6% additional computation, while they are compatible with each other to further improve the performance. We also find that involving computation reusing slightly hurts the accuracy at the 1st exit, which may be attributed to the compromise made by the first Transformer for downstream models. However, once the early-termination is adopted, this difference only results in trivial disadvantage when the computational budget is very small, as shown in Figure 8. DVT outperforms the baseline significantly in most cases.

Table 4: Effects of feature (F) and relationship (R) reuse. The percentages in brackets denote the additional computation compared to baselines involved by the reuse mechanisms.

Reuse		1st Exit (7x7)		2nd Exit (10x10)		3rd Exit (14x14)	
F	R	Top-1 acc.	GFLOPs	Top-1 acc.	GFLOPs	Top-1 acc.	GFLOPs
		70.33%	0.47	73.54%	1.37	76.74%	3.15
✓		69.42%	0.47	75.31%	$1.43_{(4.4\%)}$	79.21%	$3.31_{(5.1\%)}$
	✓	69.03%	0.47	75.34%	$1.41_{(2.9\%)}$	78.86%	$3.34_{(6.0\%)}$
✓	✓	69.04%	0.47	**75.65%**	$1.46_{(6.6\%)}$	**80.00%**	$3.50_{(11.1\%)}$

Design choices for the reuse mechanisms. Here we study the design of the feature and relationship reuse mechanisms. For experimental efficiency, we consider a two-exit DVT based on T2T-ViT-12 using 7x7 and 10x10 tokens, while enlarge the batch size and the initial learning rate by 4 times. Such a training setting slightly degrades the accuracy of DVT, but it is still reliable to reflect the difference between different design variants. We deactivate early-termination, and report the performance of each exit. Notably, as the FLOPs of 1st exit remain unchanged (i.e., 0.47G), we do not present it.

Table 5: Ablation studies for feature reuse.

Ablation	1st Exit (7x7) Top-1 acc.	2nd Exit (10x10) Top-1 acc.	GFLOPs
w/o reuse	**70.08%**	73.61%	1.37
Layer-wise feature reuse	69.84%	74.31%	1.43
Reuse classification token	69.79%	74.70%	1.43
Remove $f_l(\cdot), l \geq 2$	69.33%	74.73%	1.38
Remove LN in $f_l(\cdot)$	69.63%	75.05%	1.42
Ours	69.44%	**75.23%**	1.43

Table 6: Ablation studies for relationship reuse.

Ablation	1st Exit (7x7) Top-1 acc.	2nd Exit (10x10) Top-1 acc.	GFLOPs
w/o reuse	**70.08%**	73.61%	1.37
Layer-wise relationship reuse	69.63%	73.89%	1.38
Reuse final-layer relationships	69.25%	74.31%	1.39
MLP→Linear	69.20%	73.84%	1.38
Naive upsample	69.60%	73.34%	1.41
Ours	69.50%	**74.91%**	1.41

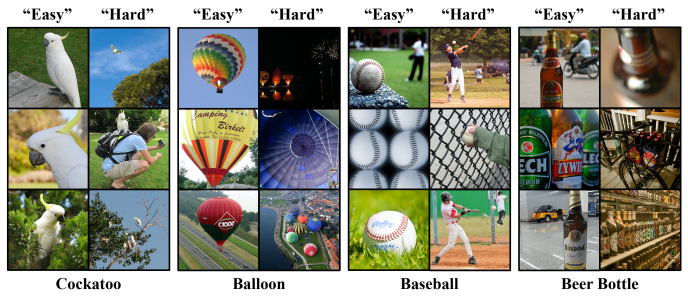

Figure 9: Visualization of the "easy" and "hard" samples in DVT.

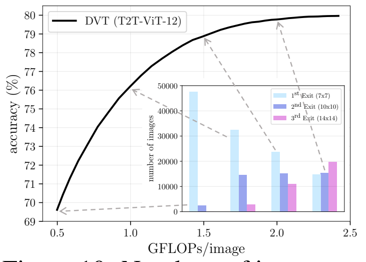

Figure 10: Numbers of images exiting at different exits with varying computational budgets.

We consider four variants of feature reuse in Table 5: (1) reusing features from the corresponding upstream layer instead of the final layer; (2) reusing classification token; (3) only performing feature reuse in the first layer of the downstream model; (4) removing the LN in $f_l(\cdot)$. One can see that taking final tokens of the upstream model and reusing them in each downstream layer are both important.

Ablation results for relationship reuse are presented in Table 6. We consider four variants as well: (1) only reusing the attention logits from the corresponding upstream layer; (2) only reusing the attention logits from the final upstream layer; (3) replacing the MLP by a linear layer in $r_l(\cdot)$; (4) adopting naive upsample operation instead of what is shown in Figure 5; The results indicate that it is beneficial to enable each downstream layer to flexibly reuse all upstream attention logits. Besides, naive upsampling significantly hurts the performance.

Early-termination. We vary the criterion for adaptive inference and report the accuracy under several computational budgets in Table 7. Two variants are considered: (1) adopting the entropy of softmax prediction to determine whether to exit [34]; (2) performing random exiting with the same exit proportion as DVT. The simple but effective confidence-based criterion achieves better performance than both of them.

Table 7: Comparisons of early-termination criterions. The accuracy under each budget is reported.

Ablation	Top-1 acc. 0.75G	1.00G	1.25G	1.50G
Randomly Exit	70.19%	71.66%	72.61%	73.59%
Entropy-based	73.41%	75.21%	77.08%	78.40%
Confidence-based (ours)	**73.70%**	**76.22%**	**77.89%**	**78.89%**

4.3 Visualization

Figure 9 shows the images that are first correctly classified at the 1st and 3rd exits of the DVT (T2T-ViT-12). The former are recognized as "easy" samples, while the later are considered to be "hard". One can observe that "easy" samples usually depict the recognition objectives in clear and canonical poses and sufficiently large resolution. On the contrary, "hard" samples may contain complex scenes and non-typical poses or only include a small part of the objects, and require a finer representation using more tokens. Figure 10 presents the numbers of images that exit at different exits when the computational budget increases. The plot shows that the accuracy of DVT is significantly improved with more images exiting later, which is achieved by changing the confidence thresholds online.

5 Conclusion

In this paper, we sought to optimally configure a proper number of tokens for each individual image in vision Transformers, and hence proposed the *Dynamic Vision Transformer* (DVT) framework. DVT processes each test input by sequentially activating multiple Transformers using increasing tokens, until an appropriate token number is reached (measured by the corresponding prediction confidence). We further introduce the feature and relationship reuse mechanisms to facilitate efficient computation reuse. Extensive experiments indicate that DVT significantly improves the computational efficiency on top of the state-of-the-art vision Transformers, both theoretically and empirically.

Acknowledgements

This work is supported in part by the National Science and Technology Major Project of the Ministry of Science and Technology of China under Grants 2018AAA0100701, the National Natural Science Foundation of China under Grants 61906106 and 62022048, Huawei Technologies Ltd. and Beijing Academy of Artificial Intelligence. In particular, we appreciate the valuable discussion with Chenghao Yang on the title of this paper.

References

[1] A. Adcock, V. Reis, M. Singh, Z. Yan, L. van der Maaten, K. Zhang, S. Motwani, J. Guerin, N. Goyal, I. Misra, L. Gustafson, C. Changhan, and P. Goyal. Classy vision. `https://github.com/facebookresearch/ClassyVision`, 2019.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *NeurIPS*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[4] Shijie Cao, Lingxiao Ma, Wencong Xiao, Chen Zhang, Yunxin Liu, Lintao Zhang, Lanshun Nie, and Zhi Yang. Seernet: Predicting convolutional neural network feature-map sparsity through low-bit quantization. In *CVPR*, pages 11216–11225, 2019.

[5] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *arXiv preprint arXiv:2103.14899*, 2021.

[6] 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.

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, pages 702–703, 2020.

[8] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*, 2021.

[9] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *ICML*, pages 248–255, 2009.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

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

[12] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021.

[13] Haoqi Fan, Yanghao Li, Bo Xiong, Wan-Yen Lo, and Christoph Feichtenhofer. Pyslowfast. `https://github.com/facebookresearch/slowfast`, 2020.

[14] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *CVPR*, pages 1039–1048, 2017.

[15] Ben 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. *arXiv preprint arXiv:2104.01136*, 2021.

[16] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

[17] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *arXiv preprint arXiv:2102.04906*, 2021.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[19] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. Transreid: Transformer-based object re-identification. *arXiv preprint arXiv:2102.04378*, 2021.

[20] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, pages 1314–1324, 2019.

[21] 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.

[22] 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 *ICLR*, 2018.

[23] Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger. Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019.

[24] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661. Springer, 2016.

[25] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[26] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *ICCV*, pages 1891–1900, 2019.

[27] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021.

[28] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *NeurIPS*, pages 2181–2191, 2017.

[29] 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. *arXiv preprint arXiv:2103.14030*, 2021.

[30] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pages 116–131, 2018.

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[32] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.

[33] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, volume 97, pages 6105–6114, 2019.

[34] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, pages 2464–2469. IEEE, 2016.

[35] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[36] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, volume 30. Curran Associates, Inc., 2017.

[38] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, pages 3–18, 2018.

[39] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *CVPR*, pages 2320–2329, 2020.

[40] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*, 2019.

[41] 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. *arXiv preprint arXiv:2102.12122*, 2021.

[42] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, pages 409–424, 2018.

[43] Yulin Wang, Zhaoxi Chen, Haojun Jiang, Shiji Song, Yizeng Han, and Gao Huang. Adaptive focus for efficient video recognition. *arXiv preprint arXiv:2105.03245*, 2021.

[44] Yulin Wang, Kangchen Lv, Rui Huang, Shiji Song, Le Yang, and Gao Huang. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. In *NeurIPS*, 2020.

[45] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[46] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

[47] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[48] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, pages 8817–8826, 2018.

[49] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *CVPR*, pages 2369–2378, 2020.

[50] Le Yang, Haojun Jiang, Ruojin Cai, Yulin Wang, Shiji Song, Gao Huang, and Qi Tian. Condensenet v2: Sparse feature reactivation for deep networks. In *CVPR*, 2021.

[51] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. *arXiv preprint arXiv:2103.11816*, 2021.

[52] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[53] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *PICCV*, pages 6023–6032, 2019.

[54] Fangao Zeng, Bin Dong, Tiancai Wang, Cheng Chen, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. *arXiv preprint arXiv:2105.03247*, 2021.

[55] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[56] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, pages 6848–6856, 2018.

[57] Moju Zhao, Kei Okada, and Masayuki Inaba. Trtr: Visual tracking with transformer. *arXiv preprint arXiv:2105.03817*, 2021.

[58] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, volume 34, pages 13001–13008, 2020.

[59] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

Appendix

A Training Details

The proposed DVT framework is implemented based on the official code of T2T-ViT[4] [52] and DeiT[5] [35] when these two models are used as backbones, respectively. The training of DVT follows exactly the same configurations as the backbones, which are also recommended by their official implementations. The details on optimizer, learning rate schedule, batch size and other hyper-parameters can be easily found in their papers or code. Note that a number of regularization and data augmentation techniques are exploited, including RandAugment [7], Random Erasing [58], Label Smoothing [32], Mixup [55], Cutmix [53], and stochastic depth [24]. We train all the models with 8 NVIDIA V100 GPUs.

Table 8: Effects when the location for performing feature reuse varies.

Ablation	1st Exit (7x7)	2nd Exit (10x10)	
	Top-1 acc.	Top-1 acc.	GFLOPs
w/o reuse	**70.08%**	73.61%	1.37
Reuse on shallow half of downstream layers	69.67%	75.02%	1.40
Reuse on deep half of downstream layers	69.94%	74.57%	1.40
Reuse on all downstream layers	69.44%	**75.23%**	1.43

Table 9: Effects when the location for performing relationship reuse varies.

Ablation	1st Exit (7x7)	2nd Exit (10x10)	
	Top-1 acc.	Top-1 acc.	GFLOPs
w/o reuse	**70.08%**	73.61%	1.37
Reuse on shallow half of downstream layers	69.72%	74.89%	1.40
Reuse on deep half of downstream layers	70.00%	73.68%	1.40
Reuse on all downstream layers	69.50%	**74.91%**	1.41

Table 10: Effects of taking attention logits from varying upstream layers.

Ablation	1st Exit (7x7)	2nd Exit (10x10)	
	Top-1 acc.	Top-1 acc.	GFLOPs
w/o reuse	**70.08%**	73.61%	1.37
Reuse relationships from shallow half of upstream layers	69.93%	74.67%	1.40
Reuse relationships from deep half of upstream layers	69.55%	74.58%	1.40
Reuse relationships from all upstream layers	69.50%	**74.91%**	1.41

B Additional Results

Which downstream layers benefit more from reuse? To shed light on the layer-wise reuse paradigm in the downstream model, we test performing feature or relationship reuse only in the shallow/deep half of downstream layers. The same experimental protocol as ablating the design of reuse mechanisms is adopted. The results are shown in Tables 8 and 9. Obviously, it is more important to reuse upstream features/relationships at the shallow layers. This phenomenon indicates that the main effects of the proposed reuse mechanisms lie in helping the first several transformer layers to rapidly extract discriminative representations or to learn accurate attention maps. The successive layers focus more on further improving the shallow features, while less on leveraging upstream information, which may have been effectively integrated into the shallow layers.

Which upstream layers contribute more to relationship reuse? In Table 10, we further test only taking the attention logits from the shallow/deep half of upstream layers in relationship reuse. One can observe that both shallow and deep relationships are important for boosting the accuracy of the

[4]`https://github.com/yitu-opensource/T2T-ViT`

[5]`https://github.com/facebookresearch/deit`

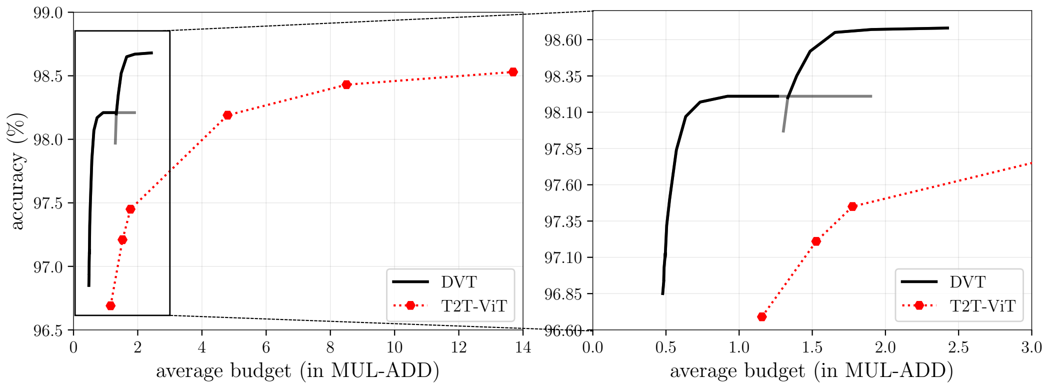

Figure 11: Top-1 accuracy v.s. GFLOPs on CIFAR-10. DVT is implemented on top of T2T-ViT-12/14.

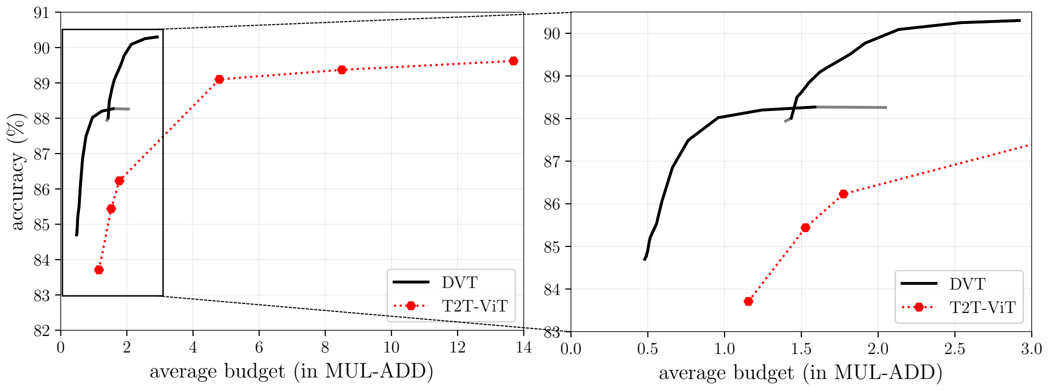

Figure 12: Top-1 accuracy v.s. GFLOPs on CIFAR-100. DVT is implemented on top of T2T-ViT-12/14.

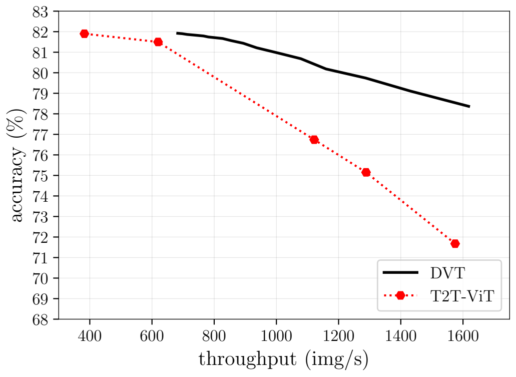

Figure 13: Top-1 accuracy v.s. throughput on ImageNet. The results are obtained on NVIDIA 2080Ti GPU with a batch size of 128.

downstream model. In addition, it is interesting that reusing more relationships only slightly improves the performance. We attribute this to the redundancy within the learned attention logits from different layers of the upstream model.

Top-1 accuracy v.s. GFLOPs curves on CIFAR-10/100 are presented in Figures 11 and 12, respectively, corresponding to the results reported in Table 3 of the paper.

Top-1 accuracy v.s. throughput curves on ImageNet are presented in Figures 13 corresponding to the results reported in Table 2 of the paper.