Sliced Recursive Transformer

Zhiqiang Shen[1,2,3], Zechun Liu[2,4], and Eric Xing[1,3]

[1] Carnegie Mellon University, Pittsburgh, USA
[2] Hong Kong University of Science and Technology, Hong Kong, China
[3] Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE
[4] Reality Labs, Meta Inc.
zhiqiangshen@cse.ust.hk,zechunliu@fb.com,epxing@cs.cmu.edu

Abstract. We present a neat yet effective recursive operation on vision transformers that can improve parameter utilization without involving additional parameters. This is achieved by sharing weights across depth of transformer networks. The proposed method can obtain a substantial gain (~2%) simply using naïve recursive operation, requires no special or sophisticated knowledge for designing principles of networks, and introduces minimal computational overhead to the training procedure. To reduce the additional computation caused by recursive operation while maintaining the superior accuracy, we propose an approximating method through multiple sliced group self-attentions across recursive layers which can reduce the cost consumption by 10~30% without sacrificing performance. We call our model **S**liced **Re**cursive **T**ransformer (SReT), a novel and parameter-efficient vision transformer design that is compatible with a broad range of other designs for efficient ViT architectures. Our best model establishes significant improvement on ImageNet-1K over state-of-the-art methods while containing fewer parameters. The proposed weight sharing mechanism by sliced recursion structure allows us to build a transformer with more than 100 or even 1000 shared layers with ease while keeping a compact size (13~15M), to avoid optimization difficulties when the model is too large. The flexible scalability has shown great potential for scaling up models and constructing extremely deep vision transformers. Code is available at https://github.com/szq0214/SReT.

1 Introduction

The architectures of transformer have achieved substantively breakthroughs recently in the fields of natural language processing (NLP) [46], computer vision (CV) [15] and speech [14,48]. In the vision area, Dosovitskiy et al. [15] introduced a vision transformer (ViT) model that splits a raw image into a patch sequence as input, and they directly adopt transformer model [46] for the image classification

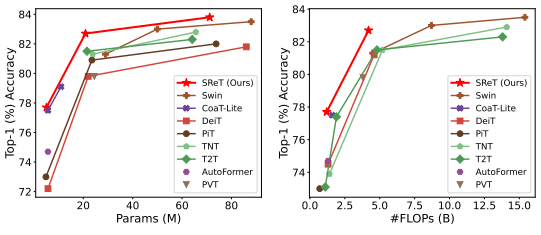


Fig. 1: Params/FLOPs vs. ImageNet-1K Acc.

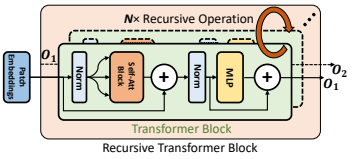


Fig. 2: Atomic Recursive Operation.

Table 1: Results using different numbers N of naïve recursive operation on ImageNet-1K dataset.

Method	Layers	#Params (M)	Top-1 Acc. (%)
DeiT-Tiny [44]	12	5.7	72.2
+ 1× naïve recursion	24	5.7	74.0
+ 2× naïve recursion	36	5.7	74.1
+ 3× naïve recursion	48	5.7	73.6

task. ViT achieved impressive results and has inspired many follow-up works. However, the benefits of a transformer often come with a large number of parameters and computational cost and it is always of great challenge to achieve the optimal trade-off between the accuracy and model complexity. In this work, we are motivated by the following question: How can we improve the parameter utilization of a vision transformer, i.e., the representation ability without increasing the model size? We observe recursive operation, as shown in Fig. 2, is a simple yet effective way to achieve this purpose. Our recursion-based vision transformer models significantly outperform state-of-the-art approaches while containing fewer parameters and FLOPs, as illustrated in Fig. 1.

Intrinsically, the classifier requires high-level abstracted features from the neural network to perform accurate classification, while the extraction of these features often requires multiple layers and deeper networks. This introduces parameter overhead into the model. Our motivation of this work stems from an interesting phenomenon of latent representation visualization. We observed that in the deep vision transformer network, the weights and activations of adjacent layers are similar with not much difference (a similar phenomenon is also discovered in [53]), which means they can be reused. The transformer with a fixed stack of distinct layers loses the inductive bias in the recurrent neural network which inspires us to share those weights in a recursive manner, forming an iterative or recursive vision transformer. Recursion can help extract stronger features without the need of increasing the parameters, and further improve the accuracy. In addition, this weight reuse or sharing strategy partially regularizes the training process by reducing the number of parameters to avoid overfitting and ill-convergence challenges, which will be discussed in the later sections.

Why do we need to introduce *sliced recursion*, i.e., the group self-attention, into transformers? (advantages and drawbacks) We usually push towards perfection on weight utilization of a network under a bounded range of parameters, thus, it can be used practically in the resource-limited circumstances like embedded devices. Recursion is a straightforward way to *compress* the feature representation in a cyclic scheme. The recursive neural networks also allow the branching of connections and structures with hierarchies. We found that it is intriguingly crucial for learning better representations on vision data in a hierarchical manner, as we will introduce in Fig. 10 of our experiments. Also, even the most simplistic recursive operation still improves the compactness of utilizing parameters without requiring to modify the transformer block

structure, unlike others [42,52,22,47,49,33,27,50], that add more parameters or involve additional fine-grained information from input [18]. However, such a recursion will incur more computational cost by its loops, namely, it *sacrifices the executing efficiency for better parameter representation utilization*. To address this shortcoming, we propose an approximating method for global self-attention through decomposing into multiple sliced group self-attentions across recursive layers, meanwhile, enjoying similar FLOPs and better representations, we also apply the spatial pyramid design to reduce the complexity of the network.

Feed-forward Networks, Recurrent Neural Networks and Recursive Neural Networks. Feed-forward networks, such as CNNs and transformers, are directed acyclic graphs (DAG). Recurrent networks (RNNs) are usually developed to process the time-series and other sequential data. Recursive network is a less frequently used term compared to other two counterparts. Recursive refers to repeating or reusing a certain piece of a network[5]. Different from RNNs that repeat the same block throughout the whole network, recursive network selectively repeats critical blocks for particular purposes. The recursive transformer iteratively refines its representations for all patches in the sequence. We found that, through the designed recursion into the feed-forward transformer, we can dramatically enhance feature representation especially for structured data without including additional parameters. More definitions are in Appendix.

The strong experimental results show that integrating the proposed sliced recursive operation in the transformer strike a competitive trade-off among accuracy, model size and complexity. To the best of our knowledge, there are barely existing works studying the effectiveness of recursive operation in vision transformers and proposing the approximation of self-attention method for reducing the complexity of recursive operation. We have done extensive experiments to derive a set of guidelines for the new design on vision task, and hope it is useful for future research. Moreover, since our method does not involve the sophisticated knowledge for modification of transformer block or additional input information, it is orthogonal and friendly to most of existing ViT designs and approaches.

Our contributions.

- We investigate the feasibility of leveraging recursive operation with sliced group self-attention in the vision transformers, which is a promising direction for establishing efficient transformers and has not been well-explored before. We conducted in-depth studies on the roles of recursion in transformers and conclude an effective scheme to use them for better parameter utilization.

- We provide design principles, including the concrete format and comprehensive comparison to variants of SReT architectures, computational equivalency analysis, modified distillation, etc., in hope of enlightening future studies in compact transformer design and optimization.

- We verify our method across a variety of scenarios, including vision transformer, all-MLP architecture of transformer variant, and neural machine translation (NMT) using transformers. Our model outperforms the state-of-the-art methods by a significant margin with fewer parameters.

[5] In a broader sense, the recurrent neural network is a type of recursive neural network.

2 Related Work

(i) **Transformer** [46] was originally designed for natural language processing tasks and has been the dominant approach [13,51,36,8,31] in this field. Recently, Vision Transformer (ViT) [15] demonstrates that such multi-head self attention blocks can completely replace convolutions and achieve competitive performance on image classification. While it relied on pre-training on large amounts of data and transferring to downstream datasets. DeiT [44] explored the training strategies and various data augmentation on ViT models, to train them on ImageNet-1K directly. Basically, DeiT can be regarded as a framework of ViT backbone + massive data augmentation + hyper-parameter tuning + hard distillation with tokens. After that, many extensions and variants of ViT models have emerged on image classification task, such as Bottleneck Transformer [42], Multimodal Transformer [20], Tokens-to-Token Transformer [52], Spatial Pyramid Transformer [22,47], Class-Attention Transformer [45], Transformer in Transformer [18], Convolution Transformer [49], Shifted Windows Transformer [33], Co-Scale Conv-Attentional Transformer [50], etc. (ii) **Recursive operation** has been explored in NLP [30,11,6,5,7,25,10] and vision [28,23,17,32] areas. In particular, DEQ [5] proposed to find equilibrium points via root-finding in the weight-tied feedforward models like transformers and trellis for constant memory. UT [11] presented the transformer with recurrent inductive bias of RNNs which is similar to our SReT format. However, these works ignored the complexity increased by recursive operation in designing networks. In this paper, we focus on utilizing recursion properly by approximating self-attention through multiple group self-attentions for building compact and efficient vision transformers.

3 Recursive Transformer

Vanilla Transformer Block. A basic transformer block $\mathcal{F}$ consists of a Multi-head Self Attention (MHSA), Layer Normalization (LN), Feed-forward Network (FFN), and Residual Connections (RC). It can be formulated as:

$$\mathbf{z}'_\ell = \mathrm{MHSA}\left(\mathrm{LN}\left(\mathbf{z}_{\ell-1}\right)\right) + \mathbf{z}_{\ell-1}; \mathbf{z}_\ell = \mathrm{FFN}\left(\mathrm{LN}\left(\mathbf{z}'_\ell\right)\right) + \mathbf{z}'_\ell; i.e., \mathbf{z}_\ell = \mathcal{F}_{\ell-1}(\mathbf{z}_{\ell-1}) \tag{1}$$

where $\mathbf{z}'_\ell$ and $\mathbf{z}_{\ell-1}$ are the intermediate representations. $\mathcal{F}_\ell$ indicates the transformer block at ℓ-th layer. $\ell \in \{0, 1, \ldots, L\}$ is the layer index and L is the number of hidden layers. The self-attention module is realized by the inner products with a scaling factor and a *softmax* operation, which is written as:

$$\mathrm{Attention}(Q, K, V) = \mathrm{Softmax}\left(QK^\top/\sqrt{d_k}\right)V \tag{2}$$

where Q, K, V are *query*, *key* and *value* vectors, respectively. $1/\sqrt{d_k}$ is the scaling factor for normalization. Multi-head self attention further concatenates the parallel attention layers to increase the representation ability:
$\mathrm{MHSA}(Q, K, V) = \mathrm{Concat}\left(\mathrm{head}_1, \ldots, \mathrm{head}_\mathrm{h}\right)W^O$, where $W^O \in \mathbb{R}^{hd_v \times d_{\mathrm{model}}}$.

$\text{head}_\text{i} = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$ are the projections with parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$. The FFN contains two linear layers with a GELU non-linearity [21] in between

$$\text{FFN}(x) = (\text{GELU}\,(\mathbf{z}W_1 + b_1))W_2 + b_2 \tag{3}$$

where $\mathbf{z}$ is the input. W_1, b_1, W_2, b_2 are the two linear layers' weights and biases.

Recursive Operation. In the original recursive module [41] for the language modality, the shared weights are recursively applied on a structured input which is among the complex inherent chains, so it is capable of learning deep structured knowledge. Recursive neural networks are made of architectural data and class, which is majorly proposed for model compositionality on NLP tasks. Here, we still use the sequence of patch tokens from the images as the inputs following the ViT model [15]. And, there are no additional inputs used for feeding into each recursive loop of recursive block as used on structured data. Take two loops as an example for building the network, the recursive operation can be simplified:

$$\mathbf{z}_\ell = \mathcal{F}_{\ell-1}(\mathcal{F}_{\ell-1}(\mathbf{z}_{\ell-1})) \tag{4}$$

The naïve recursive operation tends to learn a simple and trivial solution like the identity mapping by the optimizer, since the $\mathcal{F}_{\ell-1}$'s output and input are identical at the adjacent two depths (layers).

Non-linear Projection Layer (NLL). NLL is placed between two recursive operations to enable the non-linear transformation between each block's output and input, to avoid learning trivial status for these recursive blocks by forcing nonequivalence on neighboring output and input. NLL can be formulated as:

$$\text{NLL}(\mathbf{z}_{\ell-1}) = \text{MLP}\left(\text{LN}\left(\mathbf{z}'_{\ell-1}\right)\right) + \mathbf{z}'_{\ell-1} \tag{5}$$

where MLP is a multi-layer projection as FFN, but has different *mlp ratio* for hidden features. We also use residual connection in it for better representation. As shown in Table 1, more recursions will not improve accuracy without NLL.

Recursive Transformer. A recursive transformer with two loops in every block is:

$$\mathbf{z}_\ell = \text{NLL}_2(\mathcal{F}_{\ell-1}(\text{NLL}_1(\mathcal{F}_{\ell-1}(\mathbf{z}_{\ell-1})))) \tag{6}$$

where $\mathbf{z}_{\ell-1}$ and $\mathbf{z}_\ell$ are each recursive block's input and output. Different from MHSA and FFN that share parameters across all recursive operations within a block, NLL_1 and NLL_2 use the non-shared weights independently regardless of positioning within or outside the recursive blocks.

Recursive All-MLP [43] (an extension). We can formulate it as:

$$\begin{aligned}
\mathbf{U}_{*,i} &= \mathbf{X}_{*,i} + \mathbf{W}_2 * \text{GELU}\,(\mathbf{W}_1 *\ \text{LN}\ (\mathbf{X})_{*,i})\,, \\
\mathbf{Y}_{j,*} &= \mathbf{U}_{j,*} + \mathbf{W}_4 * \text{GELU}\,(\mathbf{W}_3 *\ \text{LN}\ (\mathbf{U})_{j,*})\,, \\
\mathbf{Y}_{j,*} &= \mathcal{M}_{\ell-1}(\mathcal{M}_{\ell-1}(\mathbf{X}_{*,i}))
\end{aligned} \tag{7}$$

where the first and second lines are *token-mixing* and *channel-mixing* from [43]. $\mathcal{M}_{\ell-1}$ is a MLP block, C is the hidden dimension and S is the number of non-overlapping image patches. NLL is not used here for simplicity.

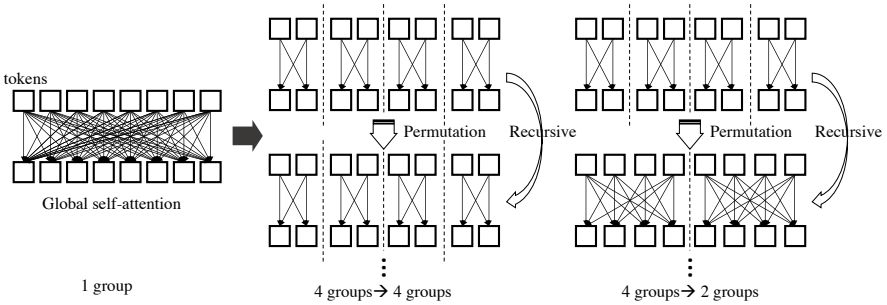


Fig. 3: Approximating global MHSA via sliced group MHSA with permutation.

Gradients in A Recursive Block. Here, we simply use explicit backpropagation through the exact operations in the forward pass like gradient descent method since SReT has no constraint to obtain the equilibrium of input-output in recursions like DEQ [5] and the number of loops can be small to control the network computation and depth. Our backward pass is more like UT [11]. In general, the gradient of the parameters in each recursive block can be:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{\mathcal{F}}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{z}^N}\frac{\partial \mathbf{z}^N}{\partial \mathbf{W}_{\mathcal{F}}} + \frac{\partial \mathcal{L}}{\partial \mathbf{z}^N}\frac{\partial \mathbf{z}^N}{\partial \mathbf{z}^{N-1}}\frac{\partial \mathbf{z}^{N-1}}{\partial \mathbf{W}_{\mathcal{F}}} + \cdots \frac{\partial \mathcal{L}}{\partial \mathbf{z}^N}\frac{\partial \mathbf{z}^N}{\partial \mathbf{z}^{N-1}}\cdots\frac{\partial \mathbf{z}^2}{\partial \mathbf{z}^1}\frac{\partial \mathbf{z}^1}{\partial \mathbf{W}_{\mathcal{F}}} \\ &= \sum_{i=1}^{N}\frac{\partial \mathcal{L}}{\partial \mathbf{z}^N}\left(\prod_{j=i}^{N-1}\frac{\partial \mathbf{z}^{j+1}}{\partial \mathbf{z}^j}\right)\frac{\partial \mathbf{z}^i}{\partial \mathbf{W}_{\mathcal{F}}}\end{aligned} \tag{8}$$

where $\mathbf{W}_{\mathcal{F}}$ is the parameters of recursive block. $\mathcal{L}$ is the objective function.

Learnable Residual Connection (LRC) for Recursive Vision Transformers. He et al. [19] studied various strategies of shortcut connections on CNNs and found that the original residual design with pre-activation performs best. Here, we found simply adding learnable coefficients on each branch of residual connection can benefit to the performance of ViT following the similar discovery of literature [29]. Formally, Eq. 1 and Eq. 5 can be reformulated as:

$$\begin{aligned}\mathbf{z}'_{\ell} &= \alpha * \mathrm{MHSA}\left(\mathrm{LN}\left(\mathbf{z}_{\ell-1}\right)\right) + \beta * \mathbf{z}_{\ell-1}; \\ \mathbf{z}_{\ell} &= \gamma * \mathrm{FFN}\left(\mathrm{LN}\left(\mathbf{z}'_{\ell}\right)\right) + \delta * \mathbf{z}'_{\ell};\end{aligned} \tag{9}$$

$$\mathrm{NLL}(\mathbf{z}_{\ell-1}) = \zeta * \mathrm{MLP}\left(\mathrm{LN}\left(\mathbf{z}'_{\ell-1}\right)\right) + \theta * \mathbf{z}'_{\ell-1} \tag{10}$$

where $\alpha, \beta, \gamma, \delta, \zeta, \theta$ are the learnable coefficients. They are initialized as 1 and trained with other model's parameters simultaneously without restrictions.

Extremely Deep Transformers. Weight-sharing mechanism allows us to build a transformer with more than 100 layers still keeping a small model. We demonstrate empirically that the proposed method can significantly simplify the optimization when the transformer is scaled up to an exaggerated number of layers.

4 Approximating Global MHSA via Multi-Group MHSA

Though recursive operation is adequate to provide better representation using the same number of parameters, the additional forward loop makes the overhead in training and inference increasing unnegligibly. To address the extra computational cost caused by recursion while maintaining the improved accuracy, we introduce an approximating method through multiple group self-attentions which is surprisingly effective in reducing FLOPs without compromising accuracy.

Approximating Global Self-Attention in SReT. As shown in Fig. 3, a regular self-attention layer can be decoupled through multiple group self-attentions in a recursion manner with similar or even smaller computational cost. In general, the number of groups in different recursion can be the same or different depending on the requirements of FLOPs and accuracy trade-off. Such a strategy will not change the number of parameters while more groups can enjoy lower FLOPs but slightly inferior performance. We empirically verified that the decoupling scheme can achieve similar performance with significantly fewer FLOPs if using proper splitting of self-attention in a tolerable scope, as shown in Appendix.

Computational Equivalency Analysis. In this subsection, we analyze the complexity of global (i.e., original) and sliced group self-attentions and compare with different values of groups in a vision transformer.

Theorem 1. *(**Equivalency of global and multiple group self-attentions with recursion on FLOPs.**) Let $\{\boldsymbol{N}_\ell, \boldsymbol{G}_\ell\} \in \mathbb{R}^1$, when $\boldsymbol{N}_\ell = \boldsymbol{G}_\ell$, **FLOPs**(1 **V-SA**)= **FLOPs**($\boldsymbol{N}_\ell \times$ Recursion with $\boldsymbol{G}_\ell \times$ **G-SAs**). The complexity $\boldsymbol{C}$ of global and group self-attentions can be calculated as: (For simplicity, here we assume #groups and vector dimensions in each recursive operation are the same.)*

$$\boldsymbol{C}_{G\text{-}SA} = \frac{\boldsymbol{N}_\ell}{\boldsymbol{G}_\ell} \times \boldsymbol{C}_{V\text{-}SA} \tag{11}$$

*where $\boldsymbol{N}_\ell$ and $\boldsymbol{G}_\ell$ are the numbers of recursion and group MHSA in layer ℓ, i.e., ℓ-th recursive block. **V-SA** and **G-SA** represent the vanilla and group MHSA.*

The proof is provided in Appendix. The insight provided by Theorem 1 is at the core of our method to control the complexity and its various benefits on better representations. Importantly, the computation of self-attention through the "slice" paralleling is equal to the vanilla self-attention. We can observe that when $\boldsymbol{N}_\ell = \boldsymbol{G}_\ell$, $\boldsymbol{C}_{V\text{-}SA} \approx \boldsymbol{C}_{G\text{-}SA}$[6] and if $\boldsymbol{N}_\ell < \boldsymbol{G}_\ell$, $\boldsymbol{C}_{G\text{-}SA} < \boldsymbol{C}_{V\text{-}SA}$, we can use this property to reduce the FLOPs in designing ViT.

Empirical observation: *When **FLOPs**(recursion + **G-SA**)$\approx$**FLOPs**(**V-SA**), **Acc.**(recursion + **G-SAs**) > **Acc.**(**V-SA**).*

We employ ex-tiny model to evaluate the performance of global self-attention and sliced group self-attention with recursion. As shown in Table 2, we empirically verify that, with the similar computation, group self-attention with recursion can obtain better accuracy than vanilla self-attention.

[6] In practice, the FLOPs of the two forms are not identical as self-attention module includes extra operations like softmax, multiplication with scale and attention values, which will be multiples by the recursive operation.

Table 2: Representation ability with global/group self-attentions.

Method	#Params (M)	FLOPs (B)	Top-1 Acc. (%)
Baseline (PiT [22])	4.9	0.7	73.0
SReT (global self-attention w/o loop)	**4.0**	0.7	73.6
SReT (group self-attentions w/ loops)	**4.0**	0.7	**74.0**

Analysis: Where is the benefit from in SReT? Theoretical analysis on recursion could further help understand the advantage behind, while it is difficult and prior literature on this always proves it empirically. Here, we provide some basic theoretical explanations from the optimization angle for better understanding this approach. One is the enhanced gradients accumulation. Let $g_t{=}\nabla_\theta f_t(\theta)$ denote the gradient, we take Adam optimizer [24] as an example, naïve parameter update is $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / \left(\sqrt{\widehat{v}_t} + \epsilon\right)$ where the gradients *w.r.t.* stochastic objective at timestep t is $g_t \leftarrow \nabla_\theta f_t\left(\theta_{t-1}\right)$, here we omit first and second moment estimate formulae. After involving recursion (here **NLL** guarantees $\widehat{m}_t^i$, $\widehat{v}_t^i$'s discrepancy), the new updating is: $\theta_t \leftarrow \theta_{t-1} - \sum_{i=1}^{\boldsymbol{N}} \alpha \cdot \widehat{m}_t^i / \left(\sqrt{\widehat{v}_t^i} + \epsilon\right)$ where $\boldsymbol{N}$ is the number of recursion loops. Basically, recursion enables more updating/tuning of parameters in the same iteration, so that the learned weights are more aligned to the loss function, and the performance is naturally better.

5 Experiments

In this section, we first empirically verify the proposed SReT on image classification task with self-attention [46] and all-MLP [43] architectures, respectively. We also perform detailed ablation studies to explore the optimal hyper-parameters of our proposed network. Then, we extend it to the neural machine translation (NMT) task to further verify the generalization ability of the proposed approach. Finally, we visualize the evolution of learned coefficients in LRC and intermediate activation maps to better understand the behaviors and properties of our proposed model. Our experiments are conducted on CIAI cluster.

5.1 Datasets and Experimental Settings

(i) **ImageNet-1K** [12]: ImageNet-1K is a standard image classification dataset, which contains 1K classes with a total number of 1.2 million training images and 50K validation images. Our models are trained on this dataset solely without additional images; (ii) **IWSLT'14 German to English (De-En)** dataset [2]: It contains about 160K sentence pairs as the training set. We train and evaluate models following the protocol [1]; (iii) **WMT'14 English to German (En-De)** dataset [3]: The WMT'14 training data consists of 4.5M sentences pairs (116M English words, 110M German words). We use the same setup as [34].

Settings: Our detailed training settings and hyper-parameters are shown in Appendix. On ImageNet-1K, our backbone network is a spatial pyramid [22] architecture with stem structure following [40].

Soft distillation strategy. On vision transformer, DeiT [44] proposed to distill tokens together with hard predictions from the teacher. They stated that using one-hot label with hard distillation can achieve the best accuracy. This seems counterintuitive since soft labels can provide more subtle differences and fine-grained information of the input. In this work, through a proper distillation design, our soft label based distillation framework (one-hot label is not used) consistently obtained better performance than DeiT[7]. Our loss is a soft version of cross-entropy between teacher and student's outputs as used in [39,37,38,4]: $\mathcal{L}_{CE}(\mathcal{S}_{\mathbf{W}}) = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{P}_{\mathcal{T}_{\mathbf{W}}}(\mathbf{z}) \log \mathbf{P}_{\mathcal{S}_{\mathbf{W}}}(\mathbf{z})$, where $\mathbf{P}_{\mathcal{T}_{\mathbf{W}}}$ and $\mathbf{P}_{\mathcal{S}_{\mathbf{W}}}$ are the outputs of teacher and student, respectively. More details can be referred to Appendix.

5.2 Naïve Recursion on Transformer

In this section, we examine the effectiveness of proposed recursion using DeiT training strategies. We verify the following two fashions of recursion.

Internal and external loops. As illustrated in Fig. 4, there are two possible recursion designs on transformer networks. One is the internal loop that repeats every block separately. Another one is the external loop that cyclically executes all blocks together. Although external loop design can force the model being more compact as it shares parameters across all blocks with fewer non-shared NLL layers, we found such a structure is inflexible with limited representation ability. We conducted a comparison with 12 layers of basic transformers with 2× recursive operation and the results are: external 67.0% (3.2M) *vs.* internal 67.6% (3.0M) | 70.3% (3.9M). In the following experiments, we use the internal recursive design as our default setting.

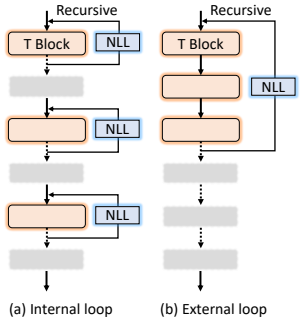


Fig. 4: Paradigms of recursive designs in transformer.

5.3 Ablation Studies

The overview of our ablation studies is shown in Table 3. The first row presents the baseline, the second group is the different structures indicated by the used factors. The last is the comparison of KD. We also verify the following designs.

[7] We observed a minor issue of soft distillation implementation in DeiT (https://github.com/facebookresearch/deit/blob/main/losses.py#L56). Basically, it is unnecessary to use *logarithm* for teacher's output (logits) according to the formulation of KL-divergence or cross-entropy. Adding *log* on both teacher and student's logits will make the results of KL to be extremely small and intrinsically negligible. We argue that soft labels can provide fine-grained information for distillation, and consistently achieve better results using soft labels in a proper way than *one-hot label + hard distillation*, as shown in Sec. 5.3.

	#Params (M)	Top-1 (%)
Baseline	5.7	72.2
Recursion $_{\text{w/o NLL}}$	3.8	72.5
Recursion + NLL	5.0	74.7
Recursion + NLL - Class Token	5.0	75.0
Recursion + NLL + LRC	5.0	75.2
Recursion + NLL + Stem	5.0	76.0
Recursion (Full Components)	5.0	**76.8**
GT+Hard Distill [44]	5.0	77.5
Soft Distill (Ours)	5.0	**77.9**

Table 3: Effectiveness of various designs on ImageNet-1K val set. Please refer to Sec. 5.3 and our Appendix for more details. In this ablation study, the backbone is `SReT-TL` model using spatial pyramid architecture.

Architecture configuration. As in Table 5, `SReT-T` is our tiny model which has *mlp ratio* = 3.6 in FFN and 4.0 for `SReT-TL`. More details about these architectures are provided in our Appendix. To examine the effectiveness of recursive operation, we conduct different loops of naïve recursion on DeiT-T. The results of accuracy curves on validation data are shown in Fig. 5 (1), we can see 2× is slightly better than 1× and the further boost is marginal, while the 1× is much faster for executing. Thus, we use this in the following experiments.

NLL configuration. NLL is a crucial factor for size and performance since the weights in it are not shared. To find an optimal trade-off between model compactness and accuracy, we explore the NLL ratios in Fig. 5 (2, 3). Generally, a larger NLL ratio can achieve better performance but the model size increases accordingly. We use 1.0 in our `SReT-T` and `SReT-TL`, and 2.0 in our `SReT-S`.

Different permutation designs and groups numbers. We explore the different permutation designs and the principle of choosing group numbers for better accuracy-FLOPs trade-off. We propose to insert permutation and inverse permutation layers to preserve the input's order information after the sliced group self-attention operation. The detailed formulation of this module, together with recursions and their result analyses are given in our Appendix.

Distillation. To examine the effectiveness of our proposed soft distillation method, we conduct the comparison of *one-hot label + hard distillation* and *soft distillation only*. The backbone network is `SReT-T`, all hyper-parameters are the same except the loss functions. The accuracy curves are shown in our Appendix. Our result 77.7% is significantly better than the baseline 77.1%.

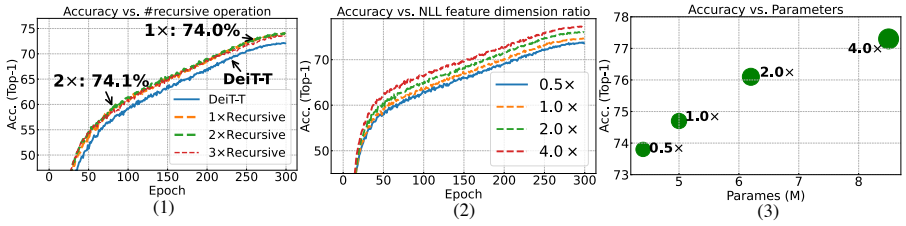


Fig. 5: A comprehensive ablation study on different design factors.

Table 4: Throughput evaluation of SReT and baselines.

DeiT-T	FLOPs: 1.3B	#Params: 5.7M	Acc.: 72.2%	Throughput: 3283.49 img/s
SReT-ExT	FLOPs: **0.7B**$^{\downarrow 46.2\%}$	#Params: **4.0M**$^{\downarrow 29.8\%}$	Acc.: **74.0%**$^{\uparrow 1.8\%}$	Throughput: **3473.43 img/s**
Swin-T	FLOPs: 4.5B	#Params: 29.0M	Acc.: 81.3%	Throughput: 1071.43 img/s
SReT-S	FLOPs: **4.2B**$^{\downarrow 6.7\%}$	#Params: **20.9M**$^{\downarrow 27.9\%}$	Acc.: **81.9%**$^{\uparrow 0.6\%}$	Throughput: **1101.84 img/s**

Throughput evaluation. In Table 4, we provide the throughput comparisons with DeiT and Swin on one NVIDIA GeForce RTX 3090 which can directly reflect the real inference speed and time consumption. We highlight that our method obtains significantly fewer params and FLOPs with better throughput.

5.4 Comparison with State-of-the-art Approaches

A summary of our main results is shown in Table 5, our `SReT-ExT` is better than PiT-T by 1.0% with 18.4%↓ parameters. `SReT-T` also outperforms DeiT-T by 3.8% with 15.8%↓ parameters and 15.4%↓ FLOPs. Distillation can help improve the accuracy by 1.6% and fine-tuning on large resolution further boosts to 79.6%. Moreover, our `SReT-S` is consistently better than state-of-the-art Swin-T, T2T, etc., on accuracy, model size and FLOPs, which demonstrates the superiority and potential of our architectures in practice.

5.5 All-MLP Architecture

MLP-Mixer [43] (Baseline), MLP-Mixer+Recursion and MLP-Mixer+Recursion +LRC: Mixer is a recently proposed plain design that is based entirely on multi-layer perceptrons (MLPs). We apply our recursive operation and LRC on MLP-Mixer to verify their generalization. Results are shown in Fig. 6 (1), our method is consistently better than the baseline using the same training protocol.

5.6 Neural Machine Translation

In this section, we compare the BLEU scores [35] of vanilla transformer [46] and ours on the WMT14 En-De and IWSLT'14 De-En (Appendix) using fairseq toolkit [16]. IWSLT'14 De-En is a relatively small dataset so the improvement is

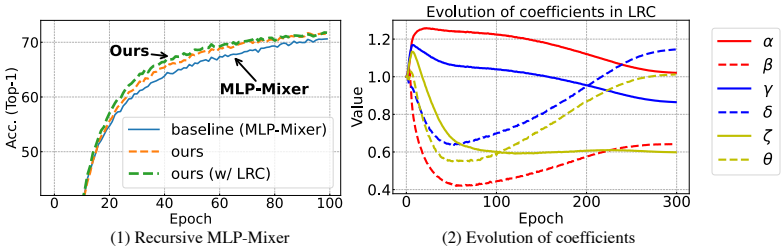


Fig. 6: (1) ImageNet-1K results on All-MLP. (2) Evolution of coefficients.

Table 5: Comparison of Top-1 (%) on ImageNet-1K with state-of-the-art methods. ⋇ denotes the model is trained without the proposed group self-attention approximation. *Fine-tuning on large resolution* is highlighted by gray color.

Method	Resolution	#Params (M)	FLOPs (B)	Top-1 (%)
DeiT-T [44]	224	5.7	1.3	72.2
PiT-T [22]	224	4.9	0.7	73.0
SReT-ExT (Ours)	224	**4.0**	0.7	**74.0**
DeiT-T [44]	224	5.7	1.3	72.2
SReT-⋇T (Ours)	224	**4.8**$^{\downarrow 15.8\%}$	1.4	**76.1**
SReT-T (Ours)	224	**4.8**	**1.1**$^{\downarrow 21.4\%}$	**76.0**
DeiT-T$_{Distill}$ [44]	224	5.7	1.3	74.5
SReT-⋇T$_{Distill}$ (Ours)	224	**4.8**	1.4	**77.7**
SReT-T$_{Distill}$ (Ours)	224	**4.8**	**1.1**$^{\downarrow 21.4\%}$	**77.6**
SReT-⋇T$_{Distill\&384\uparrow}$ (Ours)	384	4.9	6.4	79.7
SReT-T$_{Distill\&384\uparrow}$ (Ours)	384	4.9	**4.3**$^{\downarrow 32.8\%}$	79.6
DeiT-T [44]	224	5.7	1.3	72.2
AutoFormer-Tiny [9]	224	5.7	1.3	74.7
CoaT-Lite Tiny [50]	224	5.7	1.6	76.6
SReT-⋇TL (Ours)	224	**5.0**$^{\downarrow 12.3\%}$	1.4	**76.8**
SReT-TL (Ours)	224	**5.0**	**1.2**$^{\downarrow 14.3\%}$	**76.7**
SReT-⋇TL$_{Distill}$ (Ours)	224	5.0	1.4	77.9
SReT-TL$_{Distill}$ (Ours)	224	5.0	1.2	77.7
SReT-⋇TL$_{Distill\&384\uparrow}$ (Ours)	384	5.1	6.6	80.0
SReT-TL$_{Distill\&384\uparrow}$ (Ours)	384	5.1	**4.4**$^{\downarrow 33.3\%}$	79.8
ViT-B/16 [15]	384	86.0	55.4	77.9
DeiT-S [44]	224	22.1	4.6	79.8
PVT-S [47]	224	24.5	3.8	79.8
PiT-S [22]	224	23.5	2.9	80.9
T2T-ViT$_t$-14 [52]	224	21.5	5.2	80.7
TNT-S [18]	224	23.8	5.2	81.3
Swin-T [33]	224	29.0	4.5	81.3
SReT-⋇S (Ours)	224	**20.9**$^{\downarrow 27.9\%}$	4.7	**82.0**
SReT-S (Ours)	224	**20.9**	**4.2**$^{\downarrow 10.6\%}$	**81.9**
PiT-S$_{Distill}$ [22]	224	23.5	2.9	81.9
DeiT-S$_{Distill}$ [44]	224	22.1	4.6	81.2
T2T-ViT$_t$-14$_{Distill}$ [52]	224	21.5	5.2	81.7
SReT-⋇S$_{Distill}$ (Ours)	224	**20.9**	4.7	**82.8**
SReT-S$_{Distill}$ (Ours)	224	**20.9**	**4.2**$^{\downarrow 10.6\%}$	**82.7**
SReT-⋇S$_{Distill\&384\uparrow}$ (Ours)	384	**21.0**	18.5	**83.8**
SReT-⋇S$_{Distill\&512\uparrow}$ (Ours)	512	**21.3**	42.8	**84.3**

not as significant as on WMT14 En-De. The results are shown in Fig. 7, we can see our method is favorably better than the baseline. Without LRC, the model slightly converges faster, but the final accuracy is inferior to using LRC. Also, LRC makes the training process more stable, as shown in the red dashed box.

5.7 Landscape Visualizations of DeiT and Our Mixed-depth SReT

Explicit mixed-depth training. The recursive neural network enables to train the model in a mixed-depth scheme. As shown in Fig. 8 (d), the left branch is the subnetwork containing recursive blocks, while the right is the blocks without sharing the weights on depth, but their weights are re-used with the left branch. In this structure, the two branches take inputs from the same stem block. Mixed-

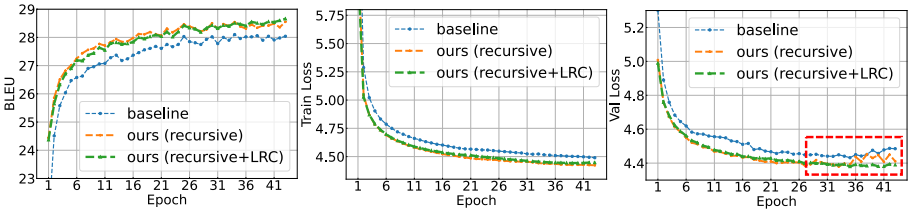


Fig. 7: Comparison of BLEU, training loss and val loss on WMT14 En-De.

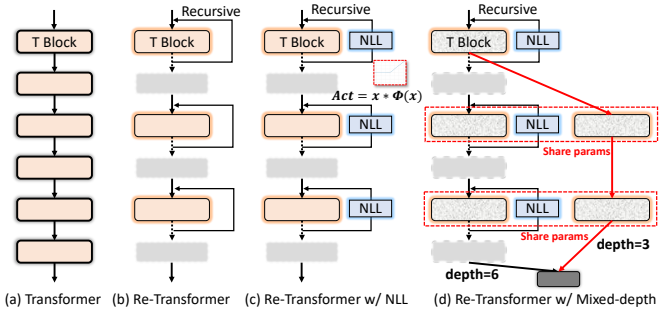
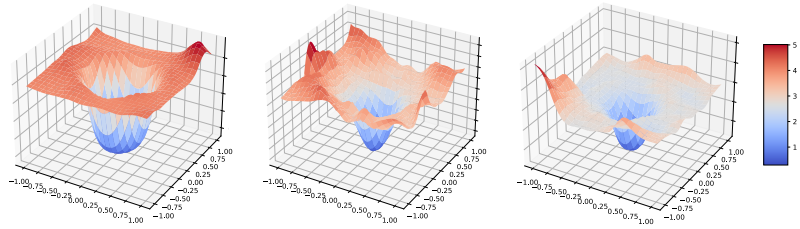


Fig. 8: Illustration of recursive transformer with different designs.

depth training offers simplified optimization by performing operations parallelly and prevents under-optimizing when the network is extremely deep.

Benefits of mixed-depth training. The spin-off benefit of sliced recursion is the feasibility of mixed-depth training, which essentially is an *explicit deep supervision* scheme as the shallow branch receives stronger supervision that is closer to the final loss layer, meanwhile, weights are shared with the deep branch.

Inspired by [26], we visualize the landscape of baseline DeiT-108 and our `SReT-108` & `SReT-108 mixed-depth` models to examine and analyze the difficulty of optimization on these three architectures. The results are illustrated in Fig. 9, we can observe that DeiT-108 is more chaotic and harder for optimization with a deeper local minimum than our mixed-depth network. This verifies the advantage of our proposed network structure for simpler optimization.



(1) DeiT-108 (2) SReT-108 (Ours) (3) SReT w/ Mixed-depth-108 (Ours)

Fig. 9: The actual optimization landscape from `DeiT-108`, our `SReT-108` and `SReT-108 mixed-depth` models.

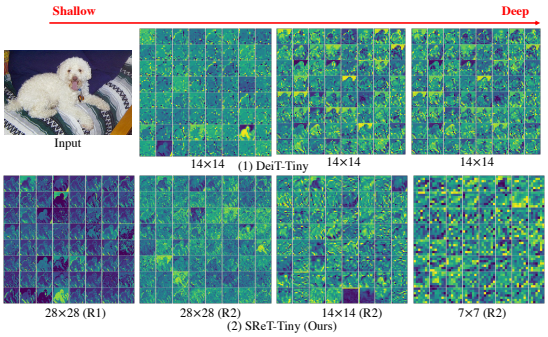


Fig. 10: Illustration of activation distributions on shallow, middle and deep layers of `DeiT-Tiny` and our `SReT-T` networks. Under each subfigure, 14×14, 28×28 and 7×7 are the resolutions of feature maps. "R1/2" indicates the index of recursive operations in each block.

5.8 Analysis and Understanding

Here, we provide two visualizations regarding LRC and learned response maps.
Evolution of LRC coefficients. As shown in Fig. 6 (2), we plot the evolution of learned coefficients in the first block. We can observe that the coefficients on the identity mapping (α, γ, ζ) first go up and then down as the training continues. This phenomenon indicates that, at the beginning of model training, the identify mapping plays a major role in the representations. After $\sim$50 epochs of training, the main branch is becoming increasingly important. Once the training is complete, in FFN and NLL, the main branch exceeds the residual connection branch while on MHSA it is the opposite. We believe this phenomenon can inspire us to design a more reasonable residual connection structure in ViT.
Learned response maps. We visualize the activation maps of DeiT-T and our `SReT-T` model at shallow and deep layers. As shown in Fig. 10, DeiT is a network with uniform resolution of feature maps (14×14). While, our spatial pyramid structure has different sizes of feature maps along with the depth of the network, i.e., the resolution of feature maps decreases when the depth increases. More interesting observations are discussed in Appendix.

6 Conclusion

It is worthwhile considering how to improve the efficiency of parameter utilization for a vision transformer with minimum overhead. In this work, we have summarized and explained several behaviors observed while training such networks. We focused on building an efficient vision transformer with a compact model size through the recursive operation, and the proposed group self-attention approximation method allows us to train in a more efficient manner with recursive transformers. We highlight such a training scheme has not been well-explored yet in previous literature. We attributed the superior performance of sliced recursive transformer to its ability of intensifying the representation quality of intermediate features. We conducted comprehensive experiments to establish the success of our method on the image classification and neural machine translation tasks, not just verifying it in the vision domain, but proving the capability to generalize for multiple modalities and architectures, such as MLP-Mixer.

References

1. https://github.com/pytorch/fairseq/blob/master/examples/translation/README.md 8
2. https://workshop2014.iwslt.org/downloads/proceeding.pdf 8
3. https://www.statmt.org/wmt14/translation-task.html 8
4. Bagherinezhad, H., Horton, M., Rastegari, M., Farhadi, A.: Label refinery: Improving imagenet classification through label progression. arXiv preprint arXiv:1805.02641 (2018) 9
5. Bai, S., Kolter, J.Z., Koltun, V.: Deep equilibrium models. In: Proceedings of the International Conference on Neural Information Processing Systems (2019) 4, 6
6. Bai, S., Kolter, J.Z., Koltun, V.: Trellis networks for sequence modeling. In: ICLR (2019) 4
7. Bai, S., Koltun, V., Kolter, J.Z.: Multiscale deep equilibrium models. In: Proceedings of the International Conference on Neural Information Processing Systems (2020) 4
8. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020) 4
9. Chen, M., Peng, H., Fu, J., Ling, H.: Autoformer: Searching transformers for visual recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021) 12
10. Chowdhury, J.R., Caragea, C.: Modeling hierarchical structures with continuous recursive neural networks. In: Proceedings of the 38th International Conference on Machine Learning. pp. 1975–1988 (2021) 4
11. Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., Kaiser, L.: Universal transformers. In: International Conference on Learning Representations (2018) 4, 6
12. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 8
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 4171–4186 (2019) 4
14. Dong, L., Xu, S., Xu, B.: Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5884–5888 (2018) 1
15. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021) 1, 4, 5, 12
16. FAIR: https://github.com/pytorch/fairseq 11
17. Guo, Q., Yu, Z., Wu, Y., Liang, D., Qin, H., Yan, J.: Dynamic recursive neural network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5147–5156 (2019) 4
18. Han, K., Xiao, A., Wu, E., Guo, J., Xu, C., Wang, Y.: Transformer in transformer. arXiv preprint arXiv:2103.00112 (2021) 3, 4, 12
19. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision. pp. 630–645 (2016) 6

20. Hendricks, L.A., Mellor, J., Schneider, R., Alayrac, J.B., Nematzadeh, A.: Decoupling the role of data, attention, and losses in multimodal transformers. arXiv preprint arXiv:2102.00529 (2021) 4
21. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016) 5
22. Heo, B., Yun, S., Han, D., Chun, S., Choe, J., Oh, S.J.: Rethinking spatial dimensions of vision transformers. arXiv preprint arXiv:2103.16302 (2021) 3, 4, 8, 9, 12
23. Kim, J., Lee, J.K., Lee, K.M.: Deeply-recursive convolutional network for image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1637–1645 (2016) 4
24. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 8
25. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019) 4
26. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp. 6391–6401 (2018) 13
27. Li, Y., Zhang, K., Cao, J., Timofte, R., Van Gool, L.: Localvit: Bringing locality to vision transformers. arXiv preprint arXiv:2104.05707 (2021) 3
28. Liang, M., Hu, X.: Recurrent convolutional neural network for object recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3367–3375 (2015) 4
29. Liu, F., Gao, M., Liu, Y., Lei, K.: Self-adaptive scaling for learnable residual structure. In: Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL) (2019) 6
30. Liu, S., Yang, N., Li, M., Zhou, M.: A recursive recurrent neural network for statistical machine translation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1491–1500 (2014) 4
31. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019) 4
32. Liu, Y., Wang, Y., Wang, S., Liang, T., Zhao, Q., Tang, Z., Ling, H.: Cbnet: A novel composite backbone network architecture for object detection. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 11653–11660 (2020) 4
33. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030 (2021) 3, 4, 12
34. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1412–1421 (2015) 8
35. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (2002) 11
36. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019) 4
37. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550 (2014) 9

38. Shen, Z., He, Z., Xue, X.: Meal: Multi-model ensemble via adversarial learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4886–4893 (2019) 9
39. Shen, Z., Liu, Z., Xu, D., Chen, Z., Cheng, K.T., Savvides, M.: Is label smoothing truly incompatible with knowledge distillation: An empirical study. In: International Conference on Learning Representations (2021) 9
40. Shen, Z., Liu, Z., Li, J., Jiang, Y.G., Chen, Y., Xue, X.: Dsod: Learning deeply supervised object detectors from scratch. In: Proceedings of the IEEE international conference on computer vision. pp. 1919–1927 (2017) 9
41. Sperduti, A., Starita, A.: Supervised neural networks for the classification of structures. IEEE Transactions on Neural Networks **8**(3), 714–735 (1997) 5
42. Srinivas, A., Lin, T.Y., Parmar, N., Shlens, J., Abbeel, P., Vaswani, A.: Bottleneck transformers for visual recognition. arXiv preprint arXiv:2101.11605 (2021) 3, 4
43. Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., Dosovitskiy, A.: Mlp-mixer: An all-mlp architecture for vision. arXiv preprint arXiv:2105.01601 (2021) 5, 8, 11
44. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. arXiv preprint arXiv:2012.12877 (2020) 2, 4, 9, 10, 12
45. Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., Jégou, H.: Going deeper with image transformers. arXiv preprint arXiv:2103.17239 (2021) 4
46. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NIPS (2017) 1, 4, 8, 11
47. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. arXiv preprint arXiv:2102.12122 (2021) 3, 4, 12
48. Wang, Y., Shi, Y., Zhang, F., Wu, C., Chan, J., Yeh, C.F., Xiao, A.: Transformer in action: a comparative study of transformer-based acoustic models for large scale speech recognition applications. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6778–6782. IEEE (2021) 1
49. Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., Zhang, L.: Cvt: Introducing convolutions to vision transformers. arXiv preprint arXiv:2103.15808 (2021) 3, 4
50. Xu, W., Xu, Y., Chang, T., Tu, Z.: Co-scale conv-attentional image transformers. arXiv preprint arXiv:2104.06399 (2021) 3, 4, 12
51. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: Xlnet: Generalized autoregressive pretraining for language understanding. Advances in Neural Information Processing Systems (2019) 4
52. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token vit: Training vision transformers from scratch on imagenet. arXiv preprint arXiv:2101.11986 (2021) 3, 4, 12
53. Zhou, D., Kang, B., Jin, X., Yang, L., Lian, X., Jiang, Z., Hou, Q., Feng, J.: Deepvit: Towards deeper vision transformer. arXiv preprint arXiv:2103.11886 (2021) 2