Supplementary Materials: 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

Appendix

In this appendix, we provide details omitted in the main text, including:

• Section 1: Proof for equivalency of global self-attention and sliced group self-attention with recursive operation on FLOPs. (Sec. 4 "Approximating Global Self-Attention via Permutation of Group/Local Self-Attentions" of the main paper.)

• Section 2: Results of SReT on ImageNet ReaL [2] and ImageNetV2 [11] datasets. (Sec. 5 "Experiments and Analysis" of the main paper.)

• Section 3: More ablation results on different permutation designs and numbers of groups when approximating global self-attention on ImageNet-1K. (Sec. 5.3 "Ablation Studies" of the main paper.)

• Section 4: Pseudocode for implementing sliced group self-attention. (Sec. 4 "Approximating Global Self-Attention via Permutation of Group/Local Self-Attentions" of the main paper.)

• Section 5: Implementation details of training on ImageNet-1K. (Sec. 5.1 "Datasets and Experimental Settings" of the main paper.)

• Section 6: Hyper-parameters setting for training language models on WMT14 En-De and IWSLT14 De-En datasets. (Sec. 5.1 "Datasets and Experimental Settings" and Sec. 5.6 "Neural Machine Translation" of the main paper.)

• Section 7: Details of our `SReT-T`, `SReT-TL`, `SReT-S` and `SReT-B` architectures. (Sec. 3 "Recursive Transformer" and Sec. 5.3. "Ablation Studies" of the main paper.)

• Section 8: Details of All-MLP structure. (Sec. 5.5 "All-MLP Architecture" of the main paper.)

• Section 9: Ablation study on different LRC designs. (Sec. 3 "Recursive Transformer" and Sec. 5.8 "Analysis and Understanding" of the main paper.)

• Section 10: Observations of Response Maps. (Sec. 5.8 "Analysis and Understanding" of the main paper.)

• Section 11: More evolution visualization of LRC coefficients on ImageNet-1K dataset. (Sec. 5.8 "Analysis and Understanding" of the main paper.)

• Section 12: Evolution visualization of LRC coefficients in language model on WMT14 En-De dataset. (Sec. 5.6 "Neural Machine Translation" and Sec. 5.8 "Analysis and Understanding" of the main paper.)
• Section 13: More ablation results on directly expanding the depth of baseline DeiT model on ImageNet-1K dataset. (Sec. 5.8 "Analysis and Understanding" of the main paper.)
• Section 14: More definitions and explanations of difference to prior arts. (Sec. 2 "Related Work" of the main paper.)

1 FLOPs Analysis

One of the key benefits of our `SReT` is to control the complexity of a recursive network. We analyze the FLOPs of global (i.e., original) and sliced group self-attentions and compare them with different circumstances of groups in a vision transformer. In this section, we provide a proof to Theorem 1 which we restate below.

Theorem 1. (***Equivalency of global self-attention and group self-attention with recursive operation 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****(*$\boldsymbol{N}_\ell\times$ *Recursive with* $\boldsymbol{G}_\ell\times$ ***G-SAs****). The complexity of regular and group self-attentions can be calculated as: (For simplicity, here we assume #groups and vector dimensions in each recursive operation are the same.)*

$$\boldsymbol{C}_{G\text{-}SA} = \frac{\boldsymbol{N}_\ell}{\boldsymbol{G}_\ell} \times \boldsymbol{C}_{V\text{-}SA} \tag{1}$$

where $\boldsymbol{N}_\ell$ *is the number of recursive operation and* $\boldsymbol{G}_\ell$ *is the number of group self-attentions in layer* ℓ*, i.e.,* ℓ*-th recursive block.* ***V-SA*** *and* ***G-SA*** *represent the vanilla and group self-attentions, respectively.*

Proof. *(Theorem* 1) The complexity $\boldsymbol{C}$ of regular self-attention can be calculated as:

$$\boldsymbol{C}_{V\text{-}SA} = \mathcal{O}(\boldsymbol{L}_\ell^2 \times \boldsymbol{D}_\ell) \tag{2}$$

where $\boldsymbol{L}_\ell$ is the sequence length and $\boldsymbol{D}_\ell$ is the dimensionality of the latent representations.

The complexity of simple recursive operation without group will be:

$$\boldsymbol{C}_{recursive} = \mathcal{O}(\boldsymbol{N}_\ell \times \boldsymbol{L}_\ell^2 \times \boldsymbol{D}_\ell) \tag{3}$$

where $\boldsymbol{N}_\ell$ is the number of recursive operation.

The complexity of sliced group self-attentions with a recursive block can be calculated as:

$$\begin{aligned} \boldsymbol{C}_{G\text{-}SA} &= \mathcal{O}(\sum_i^{\boldsymbol{N}_\ell} (\boldsymbol{g}_\ell^i \times (\frac{\boldsymbol{L}_\ell}{\boldsymbol{g}_\ell^i})^2 \times \boldsymbol{d}_\ell^i)) \\ &= \mathcal{O}(\sum_i^{\boldsymbol{N}_\ell} (\frac{\boldsymbol{L}_\ell^2}{\boldsymbol{g}_\ell^i} \times \boldsymbol{d}_\ell^i)) \end{aligned} \tag{4}$$

where $\boldsymbol{g}_\ell^i \in \{\boldsymbol{G}_\ell\}$, $\boldsymbol{d}_\ell^i \in \{\boldsymbol{D}_\ell\}$, $i = 1, \ldots, \boldsymbol{N}_\ell$.

Consider the condition of #groups $\boldsymbol{g}_\ell^i$ and vector dimension $\boldsymbol{d}_\ell^i$ in each recursive operation are the same. The complexity of group self-attentions can be re-formulated as:

$$\boldsymbol{C}_{G\text{-}SA} = \mathcal{O}(\boldsymbol{N}_\ell \times \frac{\boldsymbol{L}_\ell^2}{\boldsymbol{G}_\ell} \times \boldsymbol{D}_\ell) = \frac{\boldsymbol{N}_\ell}{\boldsymbol{G}_\ell} \times \boldsymbol{C}_{V\text{-}SA} \tag{5}$$

where $\boldsymbol{G}_\ell$ is the number of group self-attentions. When $\boldsymbol{N}_\ell = \boldsymbol{G}_\ell$, $\boldsymbol{C}_{V\text{-}SA} = \boldsymbol{C}_{G\text{-}SA}$ and if $\boldsymbol{N}_\ell < \boldsymbol{G}_\ell$, $\boldsymbol{C}_{G\text{-}SA} < \boldsymbol{C}_{V\text{-}SA}$.

2 More Results and Comparisons on ImageNet ReaL [2] and ImageNetV2 [11] Datasets

In this section, we provide results on ImageNet ReaL [2] and ImageNetV2 [11] datasets. On ImageNetV2 [11], we verify our SReT models on three metrics "Top-Images", "Matched Frequency", and "Threshold 0.7". The results are shown in Table 1, we achieve consistent improvement over DeiT on various network architectures.

Table 1. More Comparison of `SReT` on ReaL [2] and ImageNetV2 [11] datasets.

Method	Network	#Parames	FLOPs	ImageNet	ReaL	ImageNetV2 Top-images	ImageNetV2 Matched-frequency	ImageNetV2 Threshold-0.7
DeiT [17]	Tiny	5.7	1.3	72.2	80.1	74.4	59.9	68.5
SReT	Tiny	**4.8**	**1.1**	**76.0**	**83.1**	**77.9**	**64.0**	**72.8**
DeiT [17]	Tiny+Distill	5.7	1.3	74.5	82.1	77.0	62.3	71.1
SReT	Tiny+Distill	**4.8**	**1.1**	**77.6**	**84.4**	**79.6**	**65.7**	**74.2**
DeiT [17]	Small	22.1	4.6	79.8	85.7	81.0	68.1	76.4
SReT	Small	**20.9**	**4.2**	**81.9**	**86.7**	**82.8**	**70.3**	**78.1**
DeiT [17]	Small+Distill	22.1	4.6	81.2	86.8	82.5	69.7	77.5
SReT	Small+Distill	**20.9**	**4.2**	**82.7**	**88.1**	**84.0**	**72.3**	**79.9**

3 Ablation Results on Different Permutation Designs and Groups Numbers

In this section, we explore the different permutation designs and the principle of choosing group numbers for the best accuracy-FLOPs trade-off. We propose to insert an inverse permutation layer to preserve the input order information after the sliced group self-attention operation. The formulation of this operation is shown in Fig. 1 and the ablation results for this design are given in Table 2 of the first group. In the table, "P" represents the permutation layer, "I" represents the inverse permutation layer and "L" indicates that we did not involve permutation and inverse permutation in *the last stage* of models when the number of groups equals 1. We use `SReT-T` and `SReT-TL` as the base structures for the ablation of

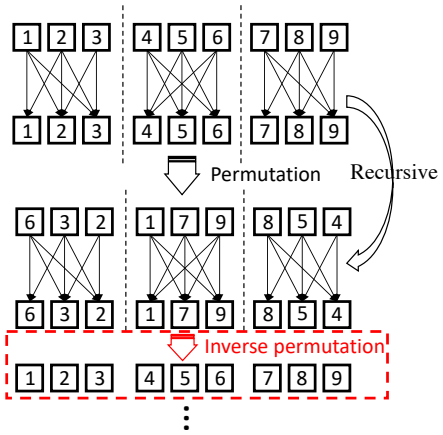

Fig. 1. Details of group self-attention with permutation designs.

different groups. In the **Groups** column of the table, we applied two loops of recursion in each recursive block according to the ablation study in Table 1 of our main text. In each pair of the square brackets, the values denote the number of groups for each recursion, and each pair of square brackets represents one stage of blocks in the spatial pyramid based backbone network. We use [8,2][4,1][1,1] as our final `SReT` structure design since it has the best trade-off on accuracy and computational cost.

Table 2. Ablation results of `SReT-T` and `SReT-TL` with different group designs.

Groups	Net	Layers	Params (M)	#FLOPs (B)	Top-1 (%)
[8,8][4,4][1,1]	P	20	4.99	1.08	75.41
[8,8][4,4][1,1]	P+I	20	4.99	1.08	75.94
[8,8][4,4][1,1]	P+I-L	20	4.99	1.08	**76.06**
[1,1][1,1][1,1]	SReT-$\divideontimes$T	20	4.76	1.38	76.07
[8,8][4,4][1,1]	SReT-T	20	4.76	1.03	75.73
[16,2][4,2][1,1]	SReT-T	20	4.76	1.01	75.79
[8,2][4,1][1,1]	SReT-T	20	4.76	1.12	75.97
[1,1][1,1][1,1]	SReT-$\divideontimes$TL	20	4.99	1.43	76.78
[8,8][4,4][1,1]	SReT-TL	20	4.99	1.08	76.06
[8,4][4,2][1,1]	SReT-TL	20	4.99	1.14	76.16
[8,2][4,1][1,1]	SReT-TL	20	4.99	1.18	76.65
[8,1][4,1][1,1]	SReT-TL	20	4.99	1.25	76.72
[16,1][14,1][1,1]	SReT-TL	20	4.99	1.24	76.56
[49,1][28,1][1,1]	SReT-TL	20	4.99	1.23	76.30

Table 3. Hyper-parameter details of conventional training.

Method	SReT-T	SReT-TL	SReT-S
Epoch	300	300	300
Batch size	1024	1024	**512**
Optimizer	AdamW	AdamW	AdamW
Learning rate	0.001	0.001	0.001
Weight decay	0.05	0.05	0.05
Warmup epochs	5	5	5
Label smoothing	0.1	0.1	0.1
Stoch. Depth	0.1	0.1	**0.2**

Table 4. Hyper-parameter details of soft distillation training.

Method	DeiT	SReT
Label	**one-hot+hard distillation**	**soft distillation**
Epoch	300	300
Batch size	1024	1024
Optimizer	AdamW	AdamW
Learning rate	0.001	0.001

Table 5. Hyper-parameter details of higher-resolution finetuning.

Method	DeiT	SReT
Resolution	384	384
Weight decay	**1e-8**	**0.0**
Learning rate	5e-6	5e-6

4 Pseudocode for Sliced Group Self-attention

The PyTorch pseudocode for implementation of our sliced group self-attention is shown in Algorithm 1.

5 Training Details on ImageNet-1K

On ImageNet-1K, we conduct experiments on three training schemes: (1) conventional training with one-hot labels; (2) distillation with soft labels from a pre-trained teacher; (3) finetuning from distilled parameters with higher resolution. Our training settings and hyper-parameters mainly follow the designs of DeiT [17]. A detailed introduction of these settings is shown in Table 3, 4 and 5 with an item-by-item comparison.

Conventional Training from Scratch with One-hot Label. As shown in Table 3, we use batch-sizes of 512/1024 for training our models and the default initial learning rate is $1e$-3, while from our experiments, larger initial lr of $2e$-3 with more warmup epochs of 30 can favorably improve the accuracy. Other settings are following [17].

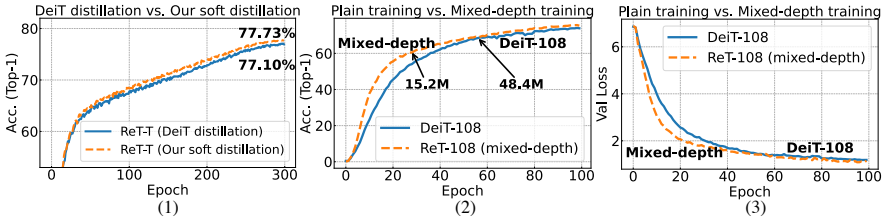

Fig. 2. A comprehensive ablation study on different design factors.

Distillation Strategy. Knowledge distillation [6] is a popular way to boost the performance of a student network. Recently, many promising results [9,15,19]

have been achieved using this technique. On vision transformer, DeiT [17] proposed to distill tokens together with hard predictions from the teacher, and it claimed 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 scheme, our soft label based distillation framework (one-hot label is not used) consistently obtained better performance than DeiT. Our loss is a soft version of cross-entropy between teacher and student's outputs as used in [12,1,13]:

$$\mathcal{L}_{CE}(\mathcal{S}_{\mathbf{W}}) = -\frac{1}{N}\sum_{i=1}^{N}\mathbf{P}_{\mathcal{T}_{\mathbf{W}}}(\mathbf{z})\log\mathbf{P}_{\mathcal{S}_{\mathbf{W}}}(\mathbf{z}) \tag{6}$$

where $\mathbf{P}_{\mathcal{T}_{\mathbf{W}}}$ and $\mathbf{P}_{\mathcal{S}_{\mathbf{W}}}$ are the outputs of teacher and student, respectively.

Distillation from Scratch. As shown in Table 4, we use soft predictions solely from RegNetY-16GF [10] as a teacher instead of *one-hot label + hard distillation* used in [17]. The ablation study on this point is provided in Fig. 2 (1) with `SReT-T`.

Spatial Pyramid (SP) Design. Pyramids [7,4] are an effective design in conventional vision tasks. The resolution of the shallow stage in a network is usually large, SP can help to redistribute the computation from shallow to deep stages of a network according to their representation ability. Here, we follow the construction principles [5] but replacing the first patch embedding layer with a *Stem block* (i.e., a stack of three 3×3 convolution layers with stride = 2) following [14].

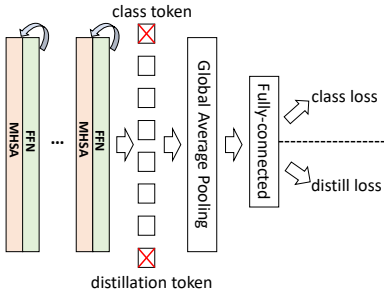

Fig. 3. Our modifications by removing class token and distillation token.

Other Small Modifications. Considering the unique properties of vision modality compared to the language, we further apply some minor modifications on our network design, some of them have been proven useful on CNNs in the vision domain, including: (i) We remove the class token and replace with a *global average pooling (GAP)* on the last output together with a fully-connected layer; (ii) We also remove the distillation token if the training process involves KD, which means we use the same feature embedding for both the ground-truth labels in standard training, and distillation with soft labels from the teacher. (iii) When fine-tuning from low resolution (224×224) to high resolution (384×384) [17], following the perspective of [15] that to increase the capacity of a model, we do not apply *weight decay* (set it as 0) during fine-tuning. Generally, the above modifications can slightly save parameters, boost the performance and significantly improve the simplicity of the whole framework. The illustration of these modifications is shown in Fig. 3.

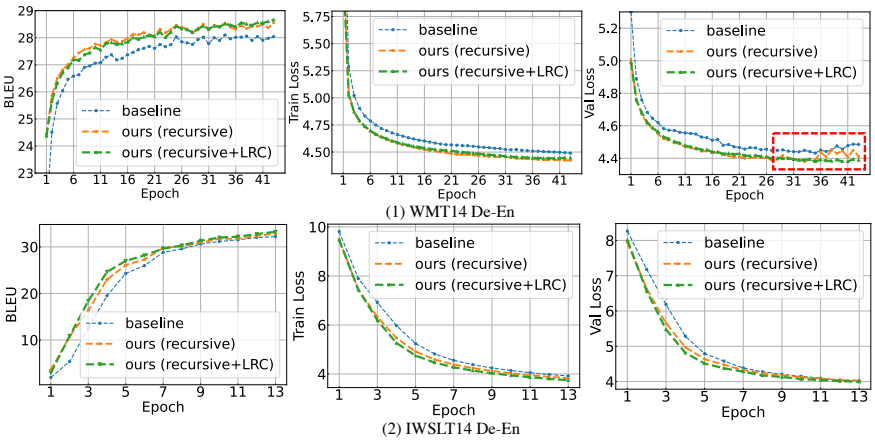

Fig. 4. Comparison of BLEU, training loss and val loss on WMT14 En-De (top) and IWSLT14 De-En datasets (bottom). The red dashed box indicates that LRC makes training more stable.

6 Hyper-parameter Settings of Language Models

We test our proposed method on two public language datasets: IWSLT14 De-En and WMT14 En-De translation tasks. We describe experimental settings in detail in Table 6.

Network Configurations. We use the Transformer [18] implemented in Fairseq [3] that shares the decoder input and output embedding as the basic NMT model.

7 Details of Our SReT Architectures

The details of our SReT-T, SReT-TL, SReT-S and SReT-B architectures are shown in Table 7. In each recursive transformer block $[[.] \times A] \times B$, A is the number of blocks with self-contained (non-shared) parameters, B is the number of recursive operations for each block. For $C\times$FFN and $D\times$NLL, C and D are the dimensions (ratios) of hidden features between the two fully-connected layers.

8 All-MLP Structure

We use B/16 in Mixer architectures [16] as our backbone network. In particular, it contains 12 layers, the patch resolution is 16×16, the hidden size C is 768, the sequence length S is 196, the MLP dimension D_C and D_S are 3072 and 384, respectively.

Table 6. Training details of our language models. The architectures we used are in Fairseq [3].

Method	IWSLT14 De-En	WMT14 En-De
arch	transformer_iwslt_de_en	transformer_wmt_en_de
share decoder input output embed	True	True
optimizer	Adam	Adam
adam-betas	(0.9, 0.98)	(0.9, 0.98)
clip-norm	0.0	0.0
learning rate	5e-4	5e-4
lr scheduler	inverse sqrt	inverse sqrt
warmup updates	4K	4K
dropout	0.3	0.3
weight decay	0.0001	0.0001
criterion	label smoothed cross-entropy	label smoothed cross-entropy
label smoothing	0.1	0.1
max tokens	4096	4096

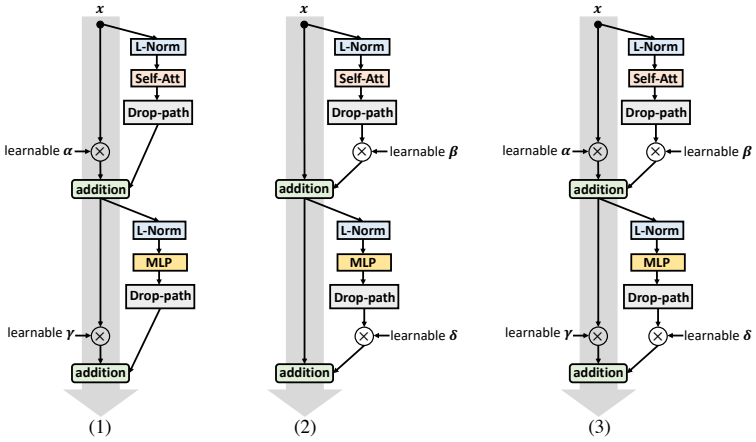

Fig. 5. Ablation study on different LRC designs.

9 Ablation Study on Different LRC Designs

In this section, we verify the effectiveness of different LRC designs as shown in Fig. 5, including: (1) learnable coefficients on the identity mapping branch; (2) learnable coefficients on the main self-attention/MLP branch; (3) our used design in the main text, i.e., including learnable coefficients on both branches.

The quantitative results of different LRC designs are shown in Table 8, we can observe that strategy (1) is slightly better than (2), while, (3) can achieve consistent improvement over (1) and (2), and it is applied in our main text. We further visualize more evolution visualizations on various layers/depths of our `SReT-TL` architecture. The results are shown in Fig. 6 and the analysis is provided in Sec. 11.

Table 7. `SReT` architectures (Input size is 3×224 × 224, sliced group self-attention is not included for simplicity.)

Layers		Output Size	SReT-T	SReT-TL
Stem	Conv-BN-ReLU	32×112×112	3×3 conv, stride 2	3×3 conv, stride 2
	Conv-BN-ReLU	64×56×56	3×3 conv, stride 2	3×3 conv, stride 2
	Conv-BN-ReLU	64×28×28	3×3 conv, stride 2	3×3 conv, stride 2
Recursive T Block (1)		64×28×28	[64-dim MHSA, 3.6×FFN/1.0×NLL] × 2 × 2	[64-dim MHSA, 4.0×FFN/1.0×NLL] × 2 × 2
Conv-Pooling Layer (1)		128×14×14	3×3 conv, stride 2, group 64	3×3 conv, stride 2, group 64
Recursive T Block (2)		128×14×14	[128-dim MHSA, 3.6×FFN/1.0×NLL] × 5 × 2	[128-dim MHSA, 4.0×FFN/1.0×NLL] × 5 × 2
Conv-Pooling Layer (2)		256×7×7	3×3 conv, stride 2, group 128	3×3 conv, stride 2, group 128
Recursive T Block (3)		256×7×7	[256-dim MHSA, 3.6×FFN/1.0×NLL] × 3 × 2	[256-dim MHSA, 4.0×FFN/1.0×NLL] × 3 × 2
Global Average Pooling		256×1×1	AdaptiveAvgPool	AdaptiveAvgPool
Linear Layer		1000		
#Params (M)			4.8 M	5.0 M
Accuracy (%)			76.1	76.8
Distilled Accuracy (%)			77.7	77.9
Finetuning Accuracy ↑384 (%)			79.7	80.0

Layers		Output Size	SReT-S	Output Size	SReT-B
Stem	Conv-BN-ReLU	63×112×112	3×3 conv, stride 2	96×112×112	3×3 conv, stride 2
	Conv-BN-ReLU	126×56×56	3×3 conv, stride 2	168×56×56	3×3 conv, stride 2
	Conv-BN-ReLU	126×28×28	3×3 conv, stride 2	336×28×28	3×3 conv, stride 2
Recursive T Block (1)		126×28×28	[126-dim MHSA, 3.0×FFN/2.0×NLL] × 2 × 2	336×28×28	[336-dim MHSA, 3.0×FFN/2.0×NLL] × 2 × 2
Conv-Pooling Layer (1)		252×14×14	3×3 conv, stride 2, group 126	672×14×14	3×3 conv, stride 2, group 336
Recursive T Block (2)		252×14×14	[252-dim MHSA, 3.0×FFN/2.0×NLL] × 5 × 2	672×14×14	[672-dim MHSA, 3.0×FFN/2.0×NLL] × 5 × 2
Conv-Pooling Layer (2)		504×7×7	3×3 conv, stride 2, group 252	1344×7×7	3×3 conv, stride 2, group 672
Recursive T Block (3)		504×7×7	[504-dim MHSA, 3.0×FFN/2.0×NLL] × 3 × 2	1344×7×7	[1344-dim MHSA, 3.0×FFN/2.0×NLL] × 3 × 2
Global Average Pooling		504×1×1	AdaptiveAvgPool	1344×1×1	AdaptiveAvgPool
Linear Layer		1000			
#Params (M)			20.9 M		71.2 M
Accuracy (%)			82.0		82.7
Distilled Accuracy (%)			82.8		83.7
Finetuning Accuracy ↑384 (%)			83.8		84.8

10 Observations of Response Maps

We have a few interesting observations on the visualizations of Fig. 8 (main text): (1) In the uniform size of transformer DeiT, information in the shallow layers is basically vague, blurry and lacks details. In contrast, the high-level layers contain stronger semantic information and are more aligned with the input. However, our model has a completely different behavior: first, in the same block but with different recursive operations, we can observe that the features are hierarchical (in Fig. 8 of main text (2)). Taken as a whole, shallow layers can capture more details like edges, shapes and contours and deep layers focus on the high-level semantic information, which is similar to CNNs. We emphasize such hierarchical representation enabled by recursion and spatial pyramid is critical for vision modality like images.

Table 8. Ablation study on different LRC designs.

Method	#Params (M)	Top-1 Acc. (%)
Baseline (`SReT-TL` w/o LRC)	5.0	74.7
on x branch (1)	5.0	75.0
on f branch (2)	5.0	74.9
on both (3)	5.0	**75.2**

11 More Evolution Visualization of LRC Coefficients on ImageNet-1K Dataset

The visualizations of coefficients evolution at different recursive blocks and layers are shown in Fig. 5. Intriguingly, we can observe in the deep layers of recursive blocks, α tends to be one stably during the whole training. Other coefficients on the identity mapping (γ and ζ) are holding fixed values that are also close to one during the training. This phenomenon indicates that the identity mapping branch tends to pass the original signal with small scaling. Moreover, it seems the contributions of the two branches have a particular proportion for the particular depth of layers.

12 Evolution Visualization of LRC Coefficients on Language Model

The visualization of coefficients evolution on the language model is shown in Fig. 7. Different from the evolution in vision transformer models, the coefficients in language model are more stable during training with small variance. Also, they are symmetrical with value one.

13 More Ablation Results on Directly Enlarging Depth of Baseline DeiT Model

In this section, we provide the results by directly expanding the depth of baseline DeiT model, as shown in Table 9. We can see deeper naïve DeiT could not bring additional gain on performance since the deeper and heavier network is usually more difficult to learn meaningful and diverse intermediate features, while our recursive operation through sharing/reusing parameters is an effective way to enlarge the depth of a transformer, meanwhile, obtaining extra improvement.

14 More Explanations and Definitions

Difference to Prior Arts: On CNNs, ShuffleNet [20] uses inerratic *shuffle* for efficient design while it is not truly random. Thus, there is no *inverse* operation involved. In contrast, our *permutation* is entirely stochastic and *inverse* is

Table 9. More ablation results on directly expanding depth of baseline DeiT model. * indicates that the total number layers of our network is 20 (recursive transformer blocks) + 10 (NLL) + 3 (image patch embeddings). Permutation and inverse permutation layers are not included.

Method	#Layers	#Params (M)	Top-1 Acc. (%)
DeiT-Tiny [17]	12	5.7	72.20
+ extend depth	24	11.55	77.35
+ extend depth	36	16.39	77.18
+ extend depth	48	21.73	75.89
Ours (SReT-S)	33*	20.90	**81.90**

crucial since self-attention is sensitive to tokens' order. The naïve group self-attention only has interaction within the window, Swin [8] addresses this using shifted windows across *different layers*. While, we solve it by integrating "slice+permutation+recursion" on the *same layer's parameters*, so each layer enables to interact with all other windows, not across layers as Swin.

Feed-forward Networks, Recurrent Neural Networks and Recursive Neural Networks. To clarify the definition of proposed recursive operation, we distinct recursive neural networks from feed-forward networks and recurrent neural networks. Feed-forward networks, such as CNNs and transformers, are directed acyclic graphs (DAG). The information path in the feed-forward processing is unidirectional, making the feed-forward networks hard to tackle the structured data with long-span correlations. Recurrent networks (RNNs) are usually developed to process the time-series and other sequential data. They output predictions based on the current input and past memory, so they are capable of processing data that contains long-term interdependent compounds like language. 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. Different from RNNs that repeat the same block throughout the whole network, recursive neural network selectively repeats critical blocks for particular purposes. The recursive transformer iteratively refines its representations for all image patches in the sequence.

Algorithm 1 PyTorch-like Code for Sliced Group MHSA with 2× Recursion.

```
# num_groups1 and num_groups2: numbers of groups in different
     recursions
# recursion: recursive indicator

class SG_Attention(nn.Module):
def __init__(self, dim, num_groups1=8, num_groups2=4, num_heads=8,
     qkv_bias=False, qk_scale=None, attn_drop=0., proj_drop=0.):
super().__init__()
self.num_heads = num_heads
# numbers of groups in different recursions
self.num_groups1 = num_groups1
self.num_groups2 = num_groups2
head_dim = dim // num_heads
self.scale = qk_scale or head_dim ** -0.5

self.qkv = nn.Linear(dim, dim * 3, bias=qkv_bias)
self.attn_drop = nn.Dropout(attn_drop)
self.proj = nn.Linear(dim, dim)
self.proj_drop = nn.Dropout(proj_drop)

def forward(self, x, recursion):
B, N, C = x.shape
if recursion == False:
num_groups = self.num_groups1
else:
num_groups = self.num_groups2
# we will not do permutation and inverse permutation if #group=1
if num_groups != 1:
idx = torch.randperm(N)
# perform permutation
x = x[:,idx,:]
# prepare for inverse permutation
inverse = torch.argsort(idx)

qkv = self.qkv(x).reshape(B, num_groups, N // num_groups, 3, self.
     num_heads, C // self.num_heads).permute(3, 0, 1, 4, 2, 5)
q, k, v = qkv[0], qkv[1], qkv[2] # make torchscript happy (cannot use
     tensor as tuple)

attn = (q @ k.transpose(-2, -1)) * self.scale
attn = attn.softmax(dim=-1)
attn = self.attn_drop(attn)

x = (attn @ v).transpose(2, 3).reshape(B, num_groups, N // num_groups,
     C)
x = x.permute(0, 3, 1, 2).reshape(B, C, N).transpose(1, 2)
if recursion == True and num_groups != 1:
# perform inverse permutation
x = x[:,inverse,:]
x = self.proj(x)
x = self.proj_drop(x)
return x
...
```

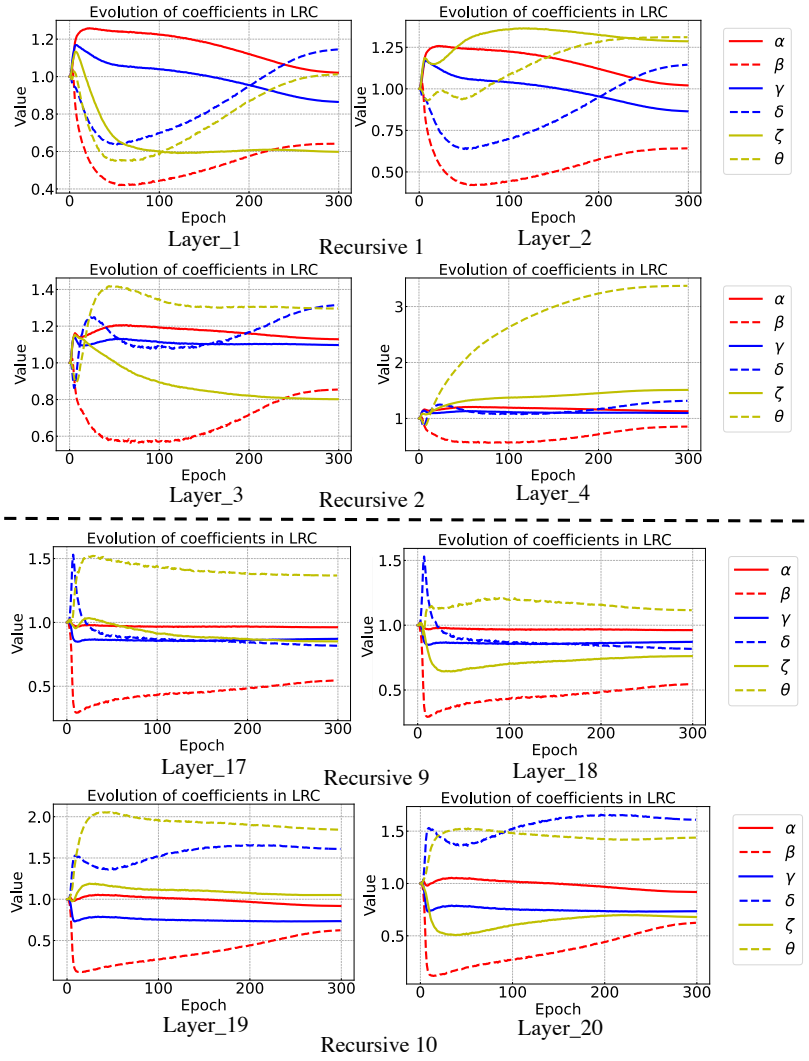

Fig. 6. Evolution of coefficients at different recursive blocks and layers.

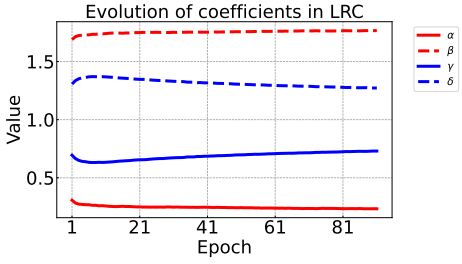

Fig. 7. Evolution of coefficients on language of WMT14 En-De dataset.

References

1. Bagherinezhad, H., Horton, M., Rastegari, M., Farhadi, A.: Label refinery: Improving imagenet classification through label progression. arXiv preprint arXiv:1805.02641 (2018) 6
2. Beyer, L., Hénaff, O.J., Kolesnikov, A., Zhai, X., Oord, A.v.d.: Are we done with imagenet? arXiv preprint arXiv:2006.07159 (2020) 1, 3
3. FAIR: https://github.com/pytorch/fairseq 7, 8
4. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence **37**(9), 1904–1916 (2015) 6
5. 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) 6
6. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015) 5
7. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 2169–2178 (2006) 6
8. 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) 11
9. Pham, H., Dai, Z., Xie, Q., Luong, M.T., Le, Q.V.: Meta pseudo labels. arXiv preprint arXiv:2003.10580 (2020) 5
10. Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10428–10436 (2020) 6
11. Recht, B., Roelofs, R., Schmidt, L., Shankar, V.: Do imagenet classifiers generalize to imagenet? In: International Conference on Machine Learning. pp. 5389–5400. PMLR (2019) 1, 3
12. 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) 6
13. 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) 6
14. 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) 6
15. Shen, Z., Savvides, M.: Meal v2: Boosting vanilla resnet-50 to 80%+ top-1 accuracy on imagenet without tricks. In: NeurIPS Workshop (2020) 5, 6
16. 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) 7
17. 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) 3, 5, 6, 11
18. 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) 7
19. Xie, Q., Luong, M.T., Hovy, E., Le, Q.V.: Self-training with noisy student improves imagenet classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10687–10698 (2020) 5

20. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6848–6856 (2018) 10