SLICED RECURSIVE TRANSFORMER

Anonymous authors
Paper under double-blind review

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 minimum 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% with minimal performance loss. We call our model **S**liced **Re**cursive **T**ransformer (SReT), which is compatible with a broad range of other designs for efficient vision transformers. Our best model establishes significant improvement on ImageNet over state-of-the-art methods while containing fewer parameters[1]. The proposed sliced recursive operation allows us to build a transformer with more than 100 or even 1000 layers effortlessly under a still small size (13~15M), to avoid difficulties in optimization when the model size is too large. The flexible scalability has shown great potential for scaling up and constructing extremely deep and large dimensionality vision transformers. Code and models will be available.

1 INTRODUCTION

The architectures of transformer have achieved substantively breakthroughs recently in the fields of natural language processing (NLP) [Vaswani et al., 2017], computer vision (CV) [Dosovitskiy et al., 2021] and speech [Dong et al., 2018, Wang et al., 2021b]. In the vision area, Dosovitskiy et al. [Dosovitskiy et al., 2021] introduced the vision transformer (ViT) method that split a raw image to a patch sequence as input and directly applied transformer model [Vaswani et al., 2017] for the image classification task. ViT achieved impressive results and has inspired many follow-up works. However, the benefits of a transformer often come with a large 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. 1 is a simple while effective way to achieve this purpose.

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 no much difference (a similar phenomenon is also discovered in [Zhou et al., 2021]), 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 of the transformer. At the same time, this weight reuse or sharing scheme also brings the effect of regularization to a certain extent, to avoid the overfitting and non-convergence issues in training, which will be discussed in the later sections.

Why do we need to introduce *recursion* into transformers? (i.e., the advantages and drawbacks) We usually push towards perfection on weight utilization of a network under a bounded range

[1]Our tiny model (~5 M) achieves Top-1 80.0% with 384×384 and 77.9% with 224×224 resolutions.

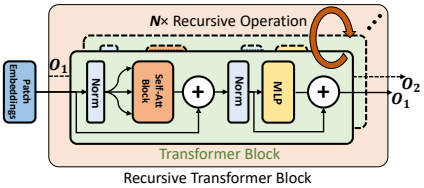


Figure 1: Atomic Recursive Operation.

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

Method	Layers	#Params (M)	Top-1 Acc. (%)
DeiT-Tiny [Touvron et al., 2020]	12	5.7	72.2
+ 1× naïve recursive	24	5.7	74.0
+ 2× naïve recursive	36	5.7	74.1
+ 3× naïve recursive	48	5.7	73.6

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 later in Fig. 6 of our experiments. Also, even the most naïve recursive operation still improves the compactness of utilizing parameters without requiring to modify the transformer block structure [Srinivas et al., 2021, Yuan et al., 2021, Heo et al., 2021, Wang et al., 2021a, Wu et al., 2021, Liu et al., 2021, Li et al., 2021, Xu et al., 2021], adding more parameters or involving additional fine-grained information from input [Han et al., 2021], but such operation will incur more computational cost, that is to say, it *sacrifices the executing efficiency for better parameter representation utilization*. To address this shortcoming, in this work 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. To clarify the definition of proposed recursive operation, we distinct recursive network 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 repeat or reuse a certain piece of a network[2]. 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. We found that, through introducing the designed recursive operation into the feed-forward transformer, we can dramatically enhance the feature representation especially for structured data without including additional parameters.

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 vision transformer design approaches.

Our contributions.

- We investigate the possibility 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 the in-depth study 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.

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

2 RELATED WORK

(i) **Transformer** [Vaswani et al., 2017] was originally designed for natural language processing tasks and has been the dominant approach [Devlin et al., 2019, Yang et al., 2019, Radford et al., 2019, Brown et al., 2020, Liu et al., 2019b] in this field. Recently, Vision Transformer (ViT) [Dosovitskiy et al., 2021] 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 [Touvron et al., 2020] 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 roadmap of ViT backbone + massive data augmentation + hyper-parameter tuning + hard distillation with token. After that, many extensions and variants of ViT models have emerged on image classification task, such as Bottleneck Transformer [Srinivas et al., 2021], Multimodal Transformer [Hendricks et al., 2021], Tokens-to-Token Transformer [Yuan et al., 2021], Spatial Pyramid Transformer [Heo et al., 2021, Wang et al., 2021a], Class-Attention Transformer [Touvron et al., 2021], Transformer in Transformer [Han et al., 2021], Convolution Transformer [Wu et al., 2021], Shifted Windows Transformer [Liu et al., 2021], Co-Scale Conv-Attentional Transformer [Xu et al., 2021], etc. (ii) **Recursive operation** has been explored in NLP [Liu et al., 2014, Dehghani et al., 2018, Bai et al., 2019a;b; 2020, Lan et al., 2019, Chowdhury & Caragea, 2021] and vision [Liang & Hu, 2015, Kim et al., 2016, Guo et al., 2019, Liu et al., 2020] areas. In particular, DEQ [Bai et al., 2019b] proposed to find equilibrium points via root-finding in the weight-tied feedforward models like transformers and trellis for constant memory. UT [Dehghani et al., 2018] 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:

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

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:

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

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: $\text{MHSA}(Q, K, V) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) W^O$, where $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$. $\text{head}_\text{i} = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$ 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 [Hendrycks & Gimpel, 2016] in between

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

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 [Sperduti & Starita, 1997] 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 [Dosovitskiy et al., 2021]. 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 as:

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

The naïve recursive operation tends to learn a 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, and avoid learning trivial status for these transformer blocks. 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 the multi-layer non-linear projection block as the FFN structure, but having different *mlp ratio* for the hidden features. We also use residual connection in it for better representation.

Recursive Transformer. A recursive transformer with two loops in every block can be written as:

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

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 [Tolstikhin et al., 2021] (an extension). $\mathcal{M}_{\ell-1}$ is a MLP block, we can formulate it as:

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

where the first and second lines are *token-mixing* and *channel-mixing* from [Tolstikhin et al., 2021]. C is the hidden dimension and S is the number of non-overlapping image patches. NLL is not used here for simplicity.

Gradients in A Recursive Block. Here, we simply use explicit backpropagation through the exact operations in the forward pass like Newton's method since SReT has no constrain to obtain the equilibrium of input-output in recursions like DEQ [Bai et al., 2019b] and the number of loops can be small to control the network computation and depth. Our backward pass is more like UT [Dehghani et al., 2018]. In general, the gradient of the parameters in each recursive block can be formulated as:

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

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

Learnable Residual Connection (LRC) for Recursive Vision Transformers. He et al. [He et al., 2016] 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 vision transformers following the similar discovery of literature [Liu et al., 2019a]. Formally, Eq. 1 and Eq. 5 can be reformulated as:

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

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

where $\alpha, \beta, \gamma, \delta, \zeta, \theta$ are the learnable coefficients. They are initialized as 1 and trained with other model's parameters simultaneously, no restrictions are added on them during training.

Extremely Deep Transformers. Weight-sharing mechanism allows us to build a transformer with more than 100 layers still with a small model size. We demonstrate empirically that the proposed method can significantly improve the optimization when the transformer is scaled up to an exaggerated number of layers, which also demonstrates that it has the regularization effect during optimization.

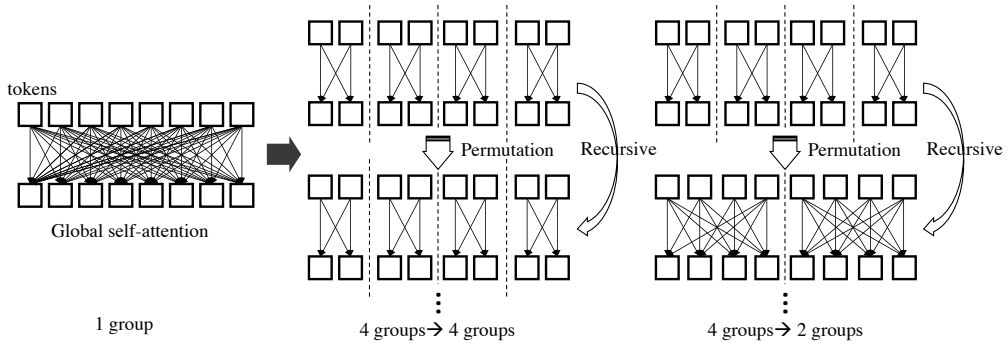


Figure 2: Approximating global self-attention via multiple sliced group-attentions with permutation.

4 Approximating Global Self-Attention via Permutations of Group/Local Self-Attentions

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 cost issue from recursive operation while maintaining the accuracy improvement, we introduce an approximating method through multiple group self-attentions which is surprisingly effective on the trade-off of accuracy and FLOPs.

Approximating Global Self-Attention in SReT. As shown in Fig. 2, 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 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.

Computational Equivalency Analysis. In this subsection we analyze the global (i.e., original) and sliced group self-attentions and compare with different values of groups in a vision transformer. Interesting observations on FLOPs and accuracy, i.e., representation ability are discussed:

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 $\boldsymbol{C}$ 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.)*

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

*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.*

The proof is provided in Appendix A. 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 $\boldsymbol{N}_\ell = G_\ell$, $\boldsymbol{C}_{V\text{-}SA} \approx \boldsymbol{C}_{G\text{-}SA}$[3] 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 transformers.

Empirical observation: *When **FLOPs**(recursive +**G-SA**)≈**FLOPs**(**V-SA**), **Acc.**(recursive + **G-SAs**) > **Acc.**(**V-SA**).*

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

Method	#Params (M)	#FLOPs (B)	Top-1 Acc. (%)
Baseline (PiT [Heo et al., 2021])	4.9	0.7	73.0
SReT (global self-attention w/o loop)	**4.0**	0.7	73.6
SReT (group self-attentions w/ $\boldsymbol{N}$ loops)	**4.0**	0.7	**74.0**

We employ the ex-tiny model to evaluate the performance of global self-attention and sliced group self-attention with recursive operation. As shown in Table 2, we empirically verify that, with the

[3]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.

similar computation, group self-attention with recursive operation can obtain better accuracy than vanilla self-attention. More comparisons are given in our experimental section.

5 Experiments and Analysis

In this section, we first empirically verify our proposed network on the image classification task with self-attention [Vaswani et al., 2017] and all-MLP [Tolstikhin et al., 2021] based architectures, respectively. We also conduct detailed ablation studies to explore the optimal hyper-parameters of our proposed approach. 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.

5.1 Datasets and Experimental Settings

(i) **ImageNet-1K** [Deng et al., 2009]: 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 train on this dataset solely without additional images; (ii) **IWSLT'14 German to English (IWSLT14 De-En)** dataset [IWS, a]: It contains about 160K sentence pairs as the training set. We train and evaluate the models following the protocol [IWS, b]; (iii) **WMT'14 English to German (WMT14 En-De)** dataset [WMT]: The WMT'14 training data consists of 4.5M sentences pairs (116M English words, 110M German words). We use the same setup as [Luong et al., 2015].

Settings: On ImageNet-1K, we mainly follow the training settings of DeiT [Touvron et al., 2020] for fair comparisons. Our backbone network is a spatial pyramid [Heo et al., 2021] architecture. On WMT14 En-De and IWSLT14 De-En, we use adam optimizer and set the initial *lr* to be *5e-4* and *inverse sqrt* scheduler. We use weight decay of *1e-4*, dropout of 0.3 and label smoothing of 0.1.

Soft distillation strategy. On vision transformer, DeiT [Touvron et al., 2020] 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 scheme, our soft label based distillation framework (one-hot label is not used) consistently obtained better performance than DeiT[4]. Our loss is a soft version of cross-entropy between teacher and student's outputs as used in [Romero et al., 2014, Bagherinezhad et al., 2018, Shen et al., 2021]: $\mathcal{L}_{CE}(\mathcal{S}_{\mathbf{W}}) = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{P}_{\mathcal{T}_{\mathbf{W}}}(\mathbf{z}) \log \mathbf{P}_{\mathcal{S}_{\mathbf{W}}}(\mathbf{z})$, 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 our Appendix E.

5.2 Naïve Recursive on Transformer

In this section, we examine the effectiveness of proposed recursive operation on ViT model using DeiT training strategies. We verify the following two fashions of recursive operation.

Internal and external loops. As illustrated in Fig. 3, there are two possible recursive designs on transformer networks. One is the internal loop that repeats every block separately. Another one is the external loop that cyclically executed all blocks together. Although external loop design can force the model being more compact as it shares parameters across all blocks, we found such 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.

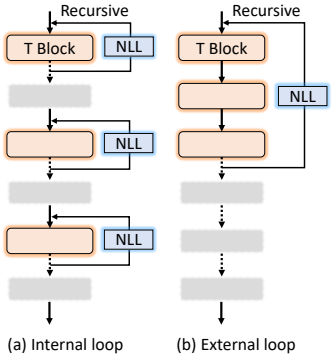


Figure 3: Recursive designs.

[4]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 of using *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 results of KL to be extremely small and intrinsically negligible. We argue that soft labels can provide fine-grained information for distillation, and consistently achieved better results using soft labels in a proper way than *one-hot label + hard distillation*, as shown in Sec. 5.3.

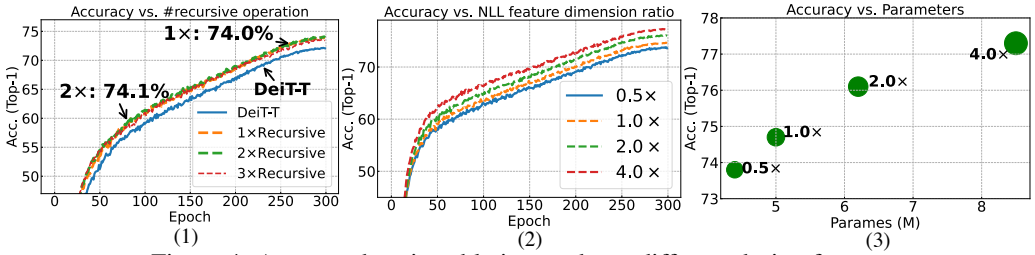


Figure 4: A comprehensive ablation study on different design factors.

5.3 Ablation Studies

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

	#Params (M)	Top-1 (%)
Baseline	5.7	72.2
Recursive + NLL	5.0	74.7
Recursive + NLL - Class Token	5.0	75.0
Recursive + NLL + Stem	5.0	76.0
Recursive + NLL + LRC	5.0	75.2
Recursive (Full components)	5.0	76.8
GT+Hard Distill [Touvron et al., 2020]	5.0	77.5
Soft Distill (Ours)	5.0	77.9

The overview of our ablation studies is shown in Table 3. The first group is the baseline and different structures, which are pointed by the used factors. We also verify the following designs.

Architecture configuration. As in Table 4, `SReT-T` is our tiny model which has *mlp ratio=3.6* in FFN and 4.0 for `SReT-TL`. More details about architectures will be given in our Appendix G. To examine the effectiveness of recursive operation, we conduct different loops of naïve recursive on DeiT-T. The results are shown in Fig. 4 (1), we can see $2\times$ is slightly better than $1\times$ and they have close accuracy, while the $1\times$ is much faster for executing. We use it in our following experiments.

NLL configuration. NLL is an important factor 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. 4 (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 information after the sliced group self-attention operation. The formulation of this module together with recursions and the result analysis are given in our Appendix C.

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 Fig. 10 (1) of Appendix. Our result 77.7% is significantly better than the baseline 77.1%.

Mixed-depth training. The spin-off benefit of recursive transformer is the feasibility of mixed-depth training, which basically is an *explicit deep supervision* fashion as the shallow branch can receive stronger supervision since it closes to the final loss layer, meanwhile, the weights are shared with the deep branch. We provide a demonstration, comparison and landscape visualization in Appendix B.

5.4 Comparison with State-of-the-art Approaches

A summary of our main results is shown in Table 4, 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 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 great potential of our architectures.

5.5 All-MLP Architecture

MLP-Mixer [Tolstikhin et al., 2021] (Baseline), MLP-Mixer + Recursive and MLP-Mixer + Recursive + LRC: Mixer is a recently proposed design that is based entirely on multi-layer perceptrons (MLPs). We apply our recursive operation and LRC on MLP-Mixer to verify the generalizability of them. Results are shown in Fig. 7 (1), our method is consistently better than the baseline using the same training protocol.

Table 4: Comparison of the Top-1 accuracy on ImageNet-1K with state-of-the-art methods. ⋇ denotes that the model is trained without the proposed group self-attention approximation.

Method	Resolution	#Params (M)	#FLOPs (B)	Top-1 (%)
DeiT-T [Touvron et al., 2020]	224	5.7	1.3	72.2
PiT-T [Heo et al., 2021]	224	4.9	0.7	73.0
SReT-ExT (Ours)	224	**4.0**	0.7	**74.0**
DeiT-T [Touvron et al., 2020]	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$_{Distill}$ [Touvron et al., 2020]	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 [Touvron et al., 2020]	224	5.7	1.3	72.2
AutoFormer-Tiny [Chen et al., 2021]	224	5.7	1.3	74.7
CoaT-Lite Tiny [Xu et al., 2021]	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 [Dosovitskiy et al., 2021]	384	86.0	55.4	77.9
DeiT-S [Touvron et al., 2020]	224	22.1	4.6	79.8
PVT-S [Wang et al., 2021a]	224	24.5	3.8	79.8
PiT-S [Heo et al., 2021]	224	23.5	2.9	80.9
T2T-ViT$_t$-14 [Yuan et al., 2021]	224	21.5	5.2	80.7
TNT-S [Han et al., 2021]	224	23.8	5.2	81.3
Swin-T [Liu et al., 2021]	224	29.0	4.5	81.3
SReT-⋇S (Ours)	224	**20.9**$^{\downarrow 27.9\%}$	4.7	**81.6**
SReT-S (Ours)	224	**20.9**	**4.2**$^{\downarrow 10.6\%}$	**81.5**
PiT-S$_{Distill}$ [Heo et al., 2021]	224	23.5	2.9	81.9
DeiT-S$_{Distill}$ [Touvron et al., 2020]	224	22.1	4.6	81.2
T2T-ViT$_t$-14$_{Distill}$ [Yuan et al., 2021]	224	21.5	5.2	81.7
SReT-⋇S$_{Distill}$ (Ours)	224	**20.9**	4.7	**82.7**
SReT-S$_{Distill}$ (Ours)	224	**20.9**	**4.2**$^{\downarrow 10.6\%}$	**82.6**
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**

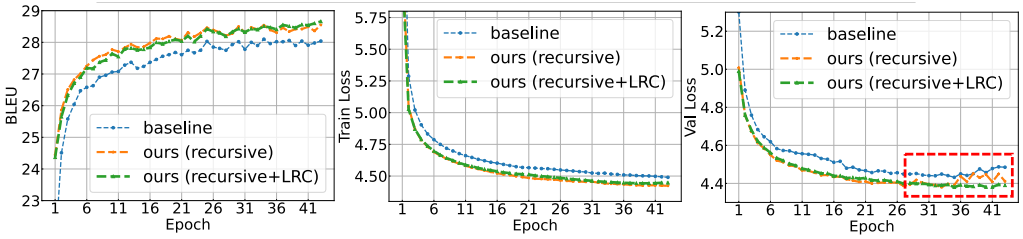


Figure 5: Comparison of BLEU, training loss and val loss on WMT14 En-De and IWSLT14 De-En (in Appendix Fig. 13) datasets. The red dashed box indicates that LRC makes training more stable.

5.6 Neural Machine Translation

In this section, we compare the BLEU scores [Papineni et al., 2002] of vanilla transformer [Vaswani et al., 2017] and ours on the WMT14 En-De and IWSLT'14 De-En datasets using fairseq toolkit [FAIR]. IWSLT'14 De-En is relatively small so the improvement is not as significant as on WMT14 En-De. The results are shown in Fig. 5, we can see without the LRC, our model can converge faster, but the final accuracy is still inferior to using LRC. Also, LRC makes the training process more stable as shown in the red dashed box.

5.7 Analysis and Understanding

Here, we provide two visualizations to better understand our trained model.
Evolution of LRC coefficients. As shown in Fig. 7 (2), we plot the evolution of learned coefficients in the first block. We can see that the coefficients on the identity mapping (α, γ, ζ) first go up and then down as the training goes on. This phenomenon indicates that, at the beginning of our model

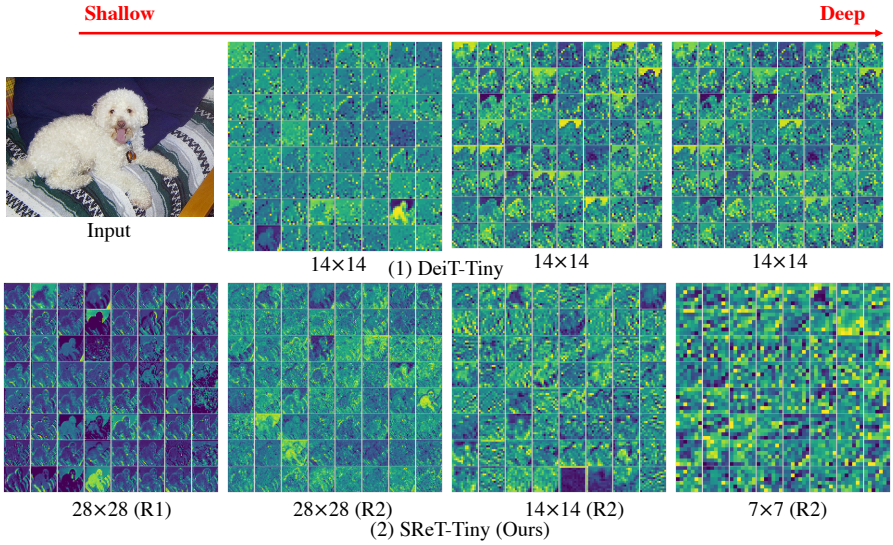


Figure 6: Illustration of activation distributions on shallow, middle and deep layers of DeiT-Tiny and our `SReT-T` networks. The input image is in the upper left corner of the first subfigure. Under each subfigure, 14×14, 28×28, 7×7 are the resolutions of feature maps. "R1&2" indicates the location of a number of recursive operations in each block. Interestingly, we observe that properties in the two models are fairly different. For DeiT, the information is from poor to rich along with the depth, while in our model, it is with hierarchies. Shallow layers focus on details and deep layers contain more semantics. Recursive operation also promotes the hierarchies. Zoom in for better view.

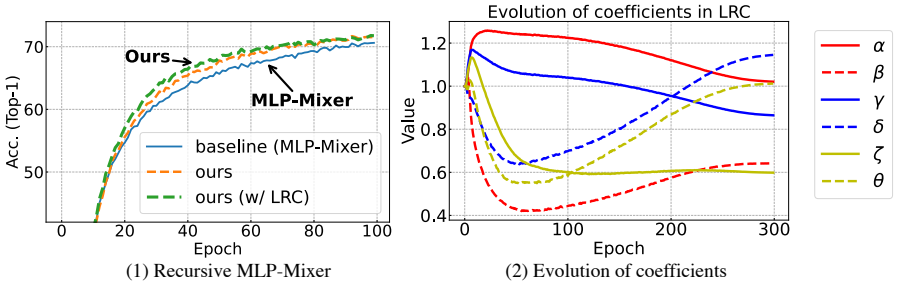


Figure 7: (1) ImageNet-1k results on All-MLP Architecture. (2) Evolution of coefficients.

training the identify mapping plays a major role. After ∼50 epochs of training, the main branch is becoming more and more important. When training is complete, in FFN and NLL, main branches exceed the residual connection while on MHSA it is the opposite. This phenomenon can inspire us to design a more reasonable residual connection in transformers.

Learned response maps. We visualize the activation maps of DeiT-Tiny and our `SReT-T` models at shallow and deep layers. As shown in Fig. 6, 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 network, i.e., the resolution of feature maps decreases as the depth increases. In each group, we visualize 64 channels in those layers. More interesting observations are introduced in Appendix J.

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 approximation method allows us to train with a more efficient manner in recursive transformers. We emphasize that such training scheme has not been explored yet in previous literature of this field. We attributed the superior performance of recursive transformer to its ability of intensifying the representation quality of intermediate features. More than solely verifying our method on the vision task, we performed comprehensive experiments to demonstrate the effectiveness on the Neural Machine Translation (NMT), showing the generalization ability for different modalities and architectures, such as MLP-Mixer.

REFERENCES

https://workshop2014.iwslt.org/downloads/proceeding.pdf. a. 6

https://github.com/pytorch/fairseq/blob/master/examples/translation/README.md. b. 6

https://www.statmt.org/wmt14/translation-task.html. 6

Hessam Bagherinezhad, Maxwell Horton, Mohammad Rastegari, and Ali Farhadi. Label refinery: Improving imagenet classification through label progression. *arXiv preprint arXiv:1805.02641*, 2018. 6, 18

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Trellis networks for sequence modeling. In *ICLR*, 2019a. 3

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2019b. 3, 4

Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. In *Proceedings of the International Conference on Neural Information Processing Systems*, 2020. 3

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 3

Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 8

Jishnu Ray Chowdhury and Cornelia Caragea. Modeling hierarchical structures with continuous recursive neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 1975–1988, 2021. 3

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2018. 3, 4

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009. 6

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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. 3

Linhao Dong, Shuang Xu, and Bo Xu. 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

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 *International Conference on Learning Representations*, 2021. 1, 3, 8

FAIR. https://github.com/pytorch/fairseq. 8, 19

Qiushan Guo, Zhipeng Yu, Yichao Wu, Ding Liang, Haoyu Qin, and Junjie Yan. Dynamic recursive neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5147–5156, 2019. 3

Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 2, 3, 8

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015. 18

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645, 2016. 4

Lisa Anne Hendricks, John Mellor, Rosalia Schneider, Jean-Baptiste Alayrac, and Aida Nematzadeh. Decoupling the role of data, attention, and losses in multimodal transformers. *arXiv preprint arXiv:2102.00529*, 2021. 3

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3

Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv preprint arXiv:2103.16302*, 2021. 2, 3, 5, 6, 8, 18

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 18

Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1637–1645, 2016. 3

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019. 3

S. Lazebnik, C. Schmid, and J. Ponce. 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)*, volume 2, pp. 2169–2178, 2006. 18

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pp. 562–570. PMLR, 2015. 16

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 6391–6401, 2018. 16

Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021. 2

Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3367–3375, 2015. 3

Fenglin Liu, Meng Gao, Yuanxin Liu, and Kai Lei. Self-adaptive scaling for learnable residual structure. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019a. 4

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 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. 3

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019b. 3

Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. Cbnet: A novel composite backbone network architecture for object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 11653–11660, 2020. 3

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. 2, 3, 8

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 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. 6

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002. 8

Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, and Quoc V Le. Meta pseudo labels. *arXiv preprint arXiv:2003.10580*, 2020. 18

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 3

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428–10436, 2020. 18

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 6, 18

Zhiqiang Shen and Marios Savvides. Meal v2: Boosting vanilla resnet-50 to 80%+ top-1 accuracy on imagenet without tricks. In *NeurIPS Workshop*, 2020. 18

Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch. In *Proceedings of the IEEE international conference on computer vision*, pp. 1919–1927, 2017. 18

Zhiqiang Shen, Zechun Liu, Dejia Xu, Zitian Chen, Kwang-Ting Cheng, and Marios Savvides. Is label smoothing truly incompatible with knowledge distillation: An empirical study. In *International Conference on Learning Representations*, 2021. 6, 18

Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997. 3

Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021. 2, 3

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021. 4, 6, 7, 20

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. 2, 3, 6, 7, 8, 18, 23

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 3

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 1, 3, 6, 8, 19

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*, 2021a. 2, 3, 8

Yongqiang Wang, Yangyang Shi, Frank Zhang, Chunyang Wu, Julian Chan, Ching-Feng Yeh, and Alex Xiao. 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, 2021b. 1

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. 2, 3

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. 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. 18

Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. *arXiv preprint arXiv:2104.06399*, 2021. 2, 3, 8

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 2019. 3

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. 2, 3, 8

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. 1

APPENDIX

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

• Section A: 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 B: A demonstration and landscape visualization of DeiT-108 and our `SReT-108 mixed-depth` models. (Sec. 3 "Recursive Transformer (Extremely Deep Transformers)" of the main paper.)

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

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

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

• Section I: Ablation study on different LRC designs. (Sec. 3 "Recursive Transformer" and Sec. 5.7 "Analysis and Understanding" of the main paper.)

• Section J: Observations of Response Maps. (Sec. 5.7 "Analysis and Understanding" of the main paper.)

• Section K: More evolution visualization of LRC coefficients on ImageNet-1K dataset. (Sec. 5.7 "Analysis and Understanding" of the main paper.)

• Section L: Evolution visualization of LRC coefficients in language model on WMT14 En-De dataset. (Sec. 5.6 "Neural Machine Translation" and Sec. 5.7 "Analysis and Understanding" of the main paper.)

• Section M: More ablation results on directly expanding the depth of baseline DeiT model on ImageNet-1K dataset. (Sec. 5.7 "Analysis and Understanding" of the main paper.)

A FLOPS ANALYSIS

One of the core 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 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.)*

$$\boldsymbol{C}_{\textit{G-SA}} = \frac{\boldsymbol{N}_\ell}{\boldsymbol{G}_\ell} \times \boldsymbol{C}_{\textit{V-SA}} \tag{12}$$

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:

$$\boldsymbol{C}_{V\text{-}SA} = \mathcal{O}(\boldsymbol{L}_\ell^2 \times \boldsymbol{D}_\ell) \tag{13}$$

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:

$$\boldsymbol{C}_{recursive} = \mathcal{O}(\boldsymbol{N}_\ell \times \boldsymbol{L}_\ell^2 \times \boldsymbol{D}_\ell) \tag{14}$$

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:

$$\begin{aligned} \boldsymbol{C}_{G\text{-}SA} &= \mathcal{O}(\sum_i^{\boldsymbol{N}_\ell} (\boldsymbol{g}_\ell^i \times (\frac{\boldsymbol{L}_\ell}{\boldsymbol{g}_\ell^i})^2 \times \boldsymbol{d}_\ell^i)) \\ &= \mathcal{O}(\sum_i^{\boldsymbol{N}_\ell} (\frac{\boldsymbol{L}_\ell^2}{\boldsymbol{g}_\ell^i} \times \boldsymbol{d}_\ell^i)) \end{aligned} \tag{15}$$

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:

$$\boldsymbol{C}_{G\text{-}SA} = \mathcal{O}(\boldsymbol{N}_\ell \times \frac{\boldsymbol{L}_\ell^2}{\boldsymbol{G}_\ell} \times \boldsymbol{D}_\ell) = \frac{\boldsymbol{N}_\ell}{\boldsymbol{G}_\ell} \times \boldsymbol{C}_{V\text{-}SA} \tag{16}$$

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 $\boldsymbol{N}_\ell < \boldsymbol{G}_\ell$, $\boldsymbol{C}_{G\text{-}SA} < \boldsymbol{C}_{V\text{-}SA}$. □

B Landscape Visualizations of Baseline DeiT and Our Mixed-depth SReT Models

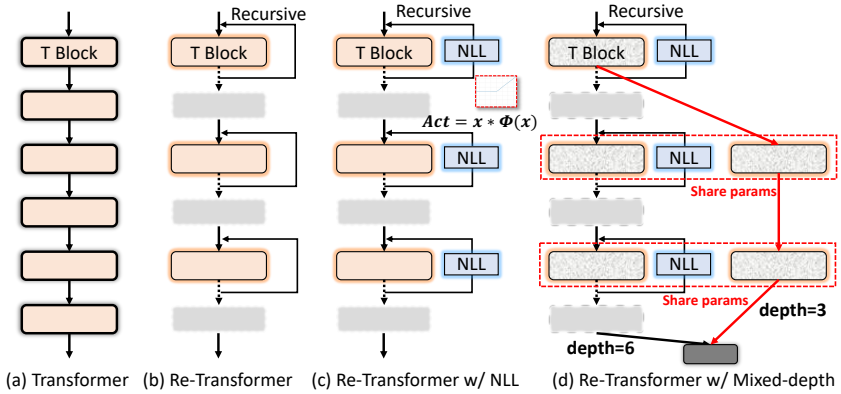


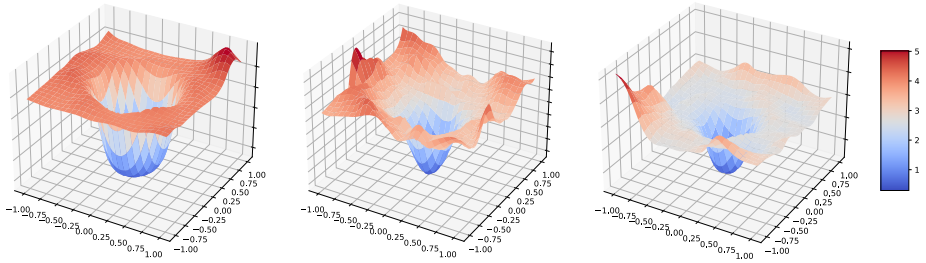
Figure 8: Illustration of recursive transformer with different designs. "NLL" indicates the non-linear projection layer within each recursive loop.

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 are 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 the inputs from the same stem block of image patches embedding. Mixed-depth training offers significant computational speedup by performing operations parallelly and prevents under-optimizing when the network is extremely deep.

Benefits of Mixed-depth Training. The spin-off benefit of recursive transformer is the feasibility of mixed-depth training, which basically is an *explicit deep supervision* fashion as the shallow branch can receive stronger supervision since it closes to the final loss layer, meanwhile, the weights are shared with the deep branch. The benefits of deep supervision have previously been demonstrated

in deeply-supervised nets [Lee et al., 2015], which utilized classifiers attached to every hidden layer, enforcing the intermediate weights to learn discriminative features. Our proposed structure performs deep supervision in a new scheme: a single classifier on top of the network provides direct supervision to two branches (deep and shallow networks) with shared weights. We verify this design by constructing an extremely deep DeiT and our model with 108 layers. As shown in Fig. 10 (2, 3), we visualize the curves of accuracy and validation loss during training. Our model converges much faster than DeiT-108 and also achieves slightly better accuracy, while our model size is only one-third of DeiT-108.

Inspired by [Li et al., 2018], 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 two architectures. The results are illustrated in Fig. 9, we can observe that DeiT-108 is more chaotic and hard for optimization with numerous local minimum than our mixed-depth network. This observation verifies the advantage of our proposed network structure for better optimization.



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

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

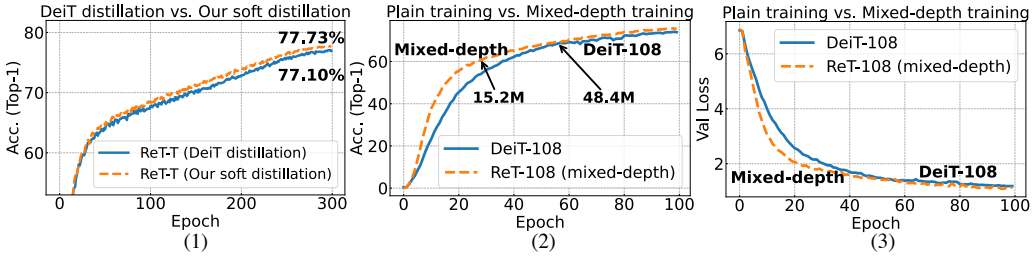


Figure 10: A comprehensive ablation study on different design factors.

C 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 better accuracy-FLOPs trade-off. We propose to insert an inverse permutation layer to preserve the input information after the sliced group self-attention operation. The formulation of this operation is shown in Fig. 11 and the ablation results for this design is given in Table 5 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 when number of groups equals 1, we use `SReT-TL` as the base structure in this group of results. In the **Groups** column of the table, we applied two loops of recursion in each recursive block according to the ablation study in 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 accuracy and computational cost trade-off.

D Pseudocode for Sliced Group Self-attention

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

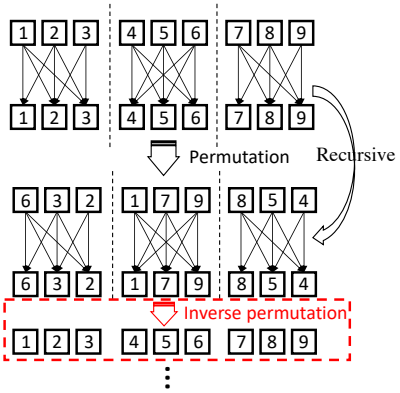


Figure 11: Different permu. designs.

Table 5: More ablation results on 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-⁎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-⁎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

Algorithm 1 PyTorch-like Code for Sliced Group Self-attention with 2× Recursive Loops.

```
# 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
   ...
```

E Training Details on ImageNet-1K

On ImageNet-1K, we conduct experimetns on three training schemes: (1) conventional training with one-hot labels; (2) distillation with soft labels from a pre-trained teacher; (3) finetuning from ditilled parameters with higher resolution. Our training settings and hyper-parameters mainly follow the

Table 6: Details of conventional training.

Method	DeiT	SReT
Epoch	300	300
Batch size	1024	512
Optimizer	AdamW	AdamW
Learning rate	0.001	0.0005
Weight decay	0.05	0.05
Warmup epochs	5	5
Label smoothing	0.1	0.1

Table 7: Details of soft distillation training.

Method	DeiT	SReT
Label	**one-hot+hard distillation**	**soft distillation**
Epoch	300	300
Batch size	1024	512
Optimizer	AdamW	AdamW
Learning rate	0.001	0.0005

Table 8: Details of higher-resolution finetuning.

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

designs of DeiT [Touvron et al., 2020]. A detailed introduction of these settings is shown in Table 6, 7 and 8 with an item-by-item comparison.

Conventional Training with One-hot Label from Scratch. As shown in Table 6, we use a batch-size of 512 for training our models and the learning rate is reduced to *5e-4* accordingly.

Distillation Strategy. Knowledge distillation [Hinton et al., 2015] is a popular way to boost the performance of a student network. Recently, many promising results [Pham et al., 2020, Shen & Savvides, 2020, Xie et al., 2020] have been achieved using this technique. On vision transformer, DeiT [Touvron et al., 2020] 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 [Romero et al., 2014, Bagherinezhad et al., 2018, Shen et al., 2021]:

$$\mathcal{L}_{CE}(\mathcal{S}_{\mathbf{W}}) = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{P}_{\mathcal{T}_{\mathbf{W}}}(\mathbf{z}) \log \mathbf{P}_{\mathcal{S}_{\mathbf{W}}}(\mathbf{z}) \tag{17}$$

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 7, we use soft predictions solely from RegNetY-16GF Radosavovic et al. [2020] as a teacher instead of *one-hot label + hard distillation* proposed in Touvron et al. [2020]. The ablation study on this point is provided in Fig. 10 (1) with `SReT-T`.

Spatial Pyramid (SP) Design. Pyramids [Lazebnik et al., 2006, He et al., 2015] 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 [Heo et al., 2021] 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 [Shen et al., 2017].

Other Small Modifications. Consider the unique properties of vision modality comparing to the language, we further apply some minor modifications on our network design, some of them have been proven useful on CNNs in 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) [Touvron et al., 2020], following the perspective of [Shen & Savvides, 2020] that to increase the capacity of a model, we do not apply *weight decay* (set it as 0) during fine-tuning.

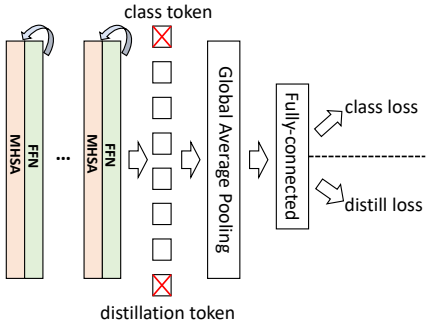


Figure 12: Our modifications by removing class token and distillation token.

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. 12.

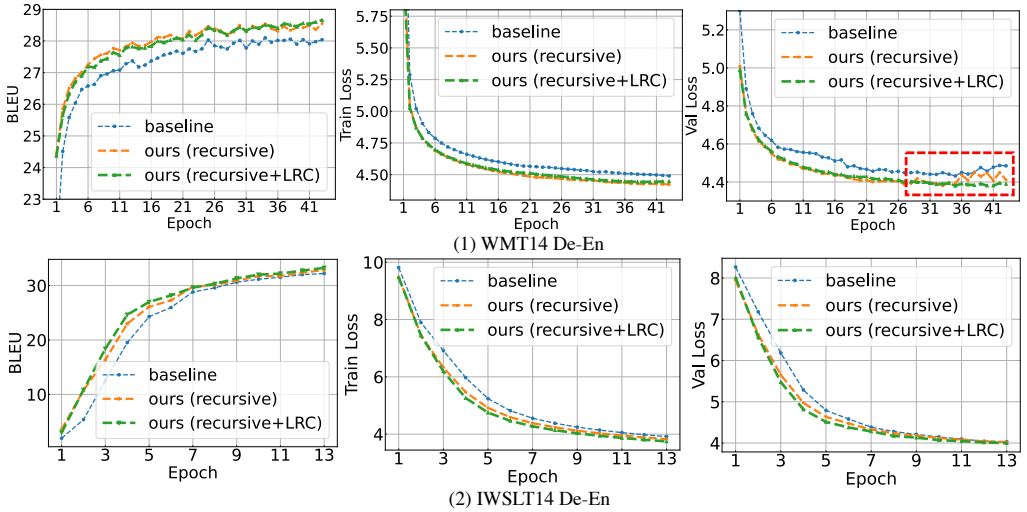


Figure 13: Comparison of BLEU, training loss and val loss on WMT14 En-De (top) and IWSLT14 De-En datasets (bottom). The red dashed box indicates that LRC makes training more stable.

F 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 9.

Network Configurations. We use the Transformer [Vaswani et al., 2017] implemented in Fairseq [FAIR] that shares the decoder input and output embedding as the basic NMT model.

Table 9: Training details of our language models. The architectures we used are in Fairseq [FAIR].

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

G Details of Our SReT Architectures

The details of our SReT-T, SReT-TL, SReT-S and SReT-B architectures are shown in Table 10. 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.

Table 10: SReT architectures (Input size is $3\times224\times224$, sliced group 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	$\left[\left[\begin{array}{c}\text{64-dim MHSA}\\ \text{3.6×FFN/1.0×NLL}\end{array}\right]\times 2\right]\times 2$	$\left[\left[\begin{array}{c}\text{64-dim MHSA}\\ \text{4.0×FFN/1.0×NLL}\end{array}\right]\times 2\right]\times 2$
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	$\left[\left[\begin{array}{c}\text{128-dim MHSA}\\ \text{3.6×FFN/1.0×NLL}\end{array}\right]\times 5\right]\times 2$	$\left[\left[\begin{array}{c}\text{128-dim MHSA}\\ \text{4.0×FFN/1.0×NLL}\end{array}\right]\times 5\right]\times 2$
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	$\left[\left[\begin{array}{c}\text{256-dim MHSA}\\ \text{3.6×FFN/1.0×NLL}\end{array}\right]\times 3\right]\times 2$	$\left[\left[\begin{array}{c}\text{256-dim MHSA}\\ \text{4.0×FFN/1.0×NLL}\end{array}\right]\times 3\right]\times 2$
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	$\left[\left[\begin{array}{c}\text{126-dim MHSA}\\ \text{3.0×FFN/2.0×NLL}\end{array}\right]\times 2\right]\times 2$	336×28×28	$\left[\left[\begin{array}{c}\text{336-dim MHSA}\\ \text{3.0×FFN/2.0×NLL}\end{array}\right]\times 2\right]\times 2$
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	$\left[\left[\begin{array}{c}\text{252-dim MHSA}\\ \text{3.0×FFN/2.0×NLL}\end{array}\right]\times 5\right]\times 2$	672×14×14	$\left[\left[\begin{array}{c}\text{672-dim MHSA}\\ \text{3.0×FFN/2.0×NLL}\end{array}\right]\times 5\right]\times 2$
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	$\left[\left[\begin{array}{c}\text{504-dim MHSA}\\ \text{3.0×FFN/2.0×NLL}\end{array}\right]\times 3\right]\times 2$	1344×7×7	$\left[\left[\begin{array}{c}\text{1344-dim MHSA}\\ \text{3.0×FFN/2.0×NLL}\end{array}\right]\times 3\right]\times 2$
Global Average Pooling		504×1×1	AdaptiveAvgPool	1344×1×1	AdaptiveAvgPool
Linear Layer		1000			
#Params (M)			20.9 M		71.2 M
Accuracy (%)			81.6		82.7
Distilled Accuracy (%)			82.7		83.7
Finetuning Accuracy ↑384 (%)			83.8		84.8

H ALL-MLP STRUCTURE

We use B/16 in Mixer architectures [Tolstikhin et al., 2021] 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.

I ABLATION STUDY ON DIFFERENT LRC DESIGNS

In this section, we verify the effectiveness of different LRC designs as shown in Fig. 14, including: (1) learnable coefficients on 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 11, 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. 15 and the analysis is provided in Sec. K.

J OBSERVATIONS OF RESPONSE MAPS

We have a few interesting observations on these visualizations of Fig. 6: (1) In the uniform size of transformer DeiT, information in the shallow layers are basically vague, blurry and lacks details, while the high-level layers contain stronger semantic information and are more aligned for the input.

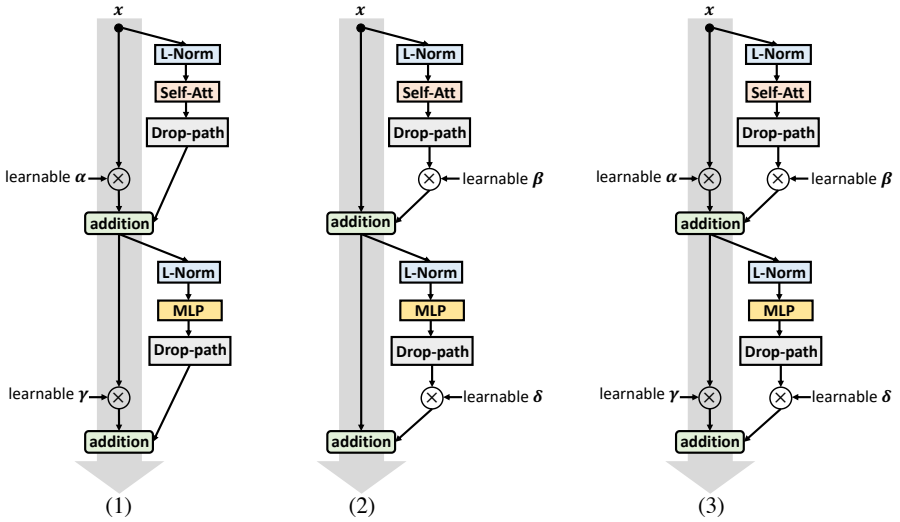


Figure 14: Ablation study on different LRC designs.

Table 11: 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**

However, our model has a completely different behavior: first, in the same block but different recursive operation, we can observe that the features are hierarchical (Fig. 6 (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 recursive and spatial pyramid is critical for vision modality like images.

K More Evolution Visualization of LRC Coefficients on ImageNet Dataset

The visualizations of coefficients evolution at different recursive blocks and layers are shown in Fig. 14. 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.

L 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 much stable during training with small variance. Also, they are symmetrical with value one.

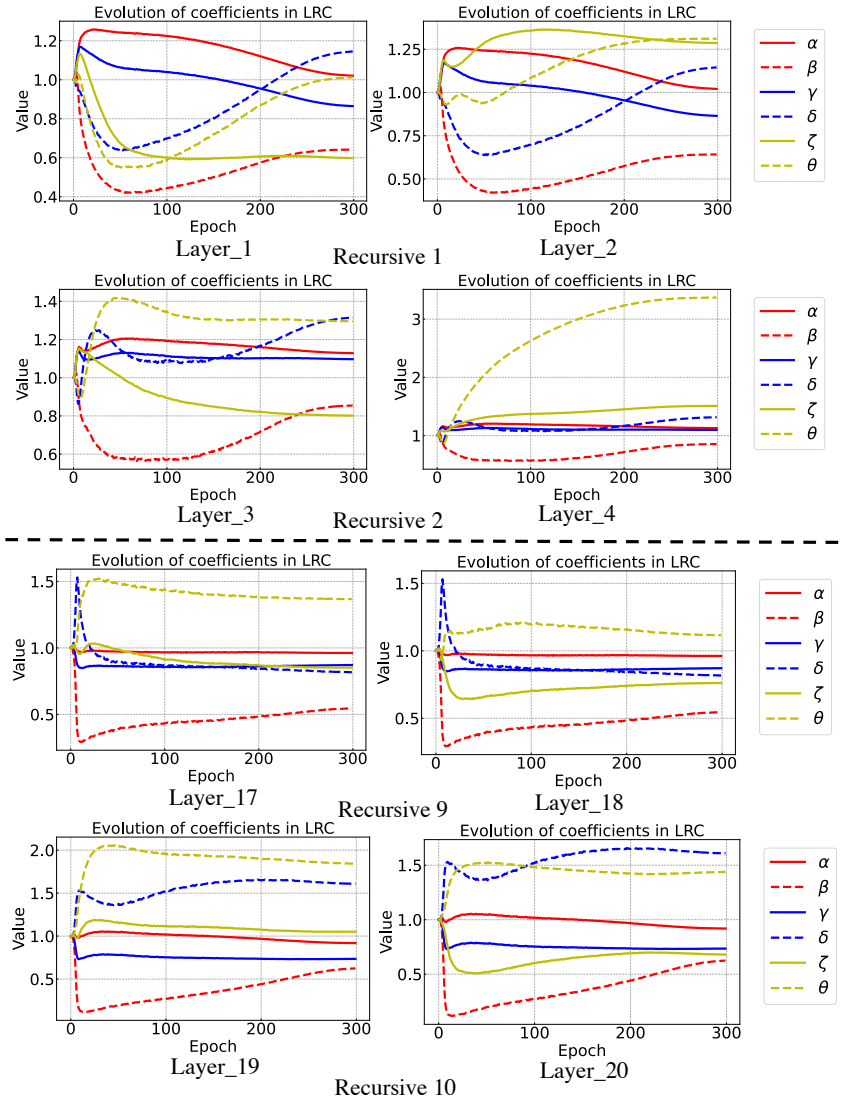


Figure 15: Evolution of coefficients at different recursive blocks and layers.

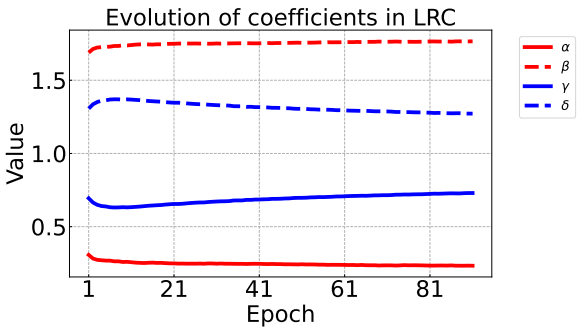


Figure 16: Evolution of coefficients on language of WMT14 En-De dataset.

Table 12: More ablation results on directly expanding depth of baseline DeiT model. * 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 Touvron et al. [2020]	12	5.7	72.2
+ 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	**81.56**

M 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 12. 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.