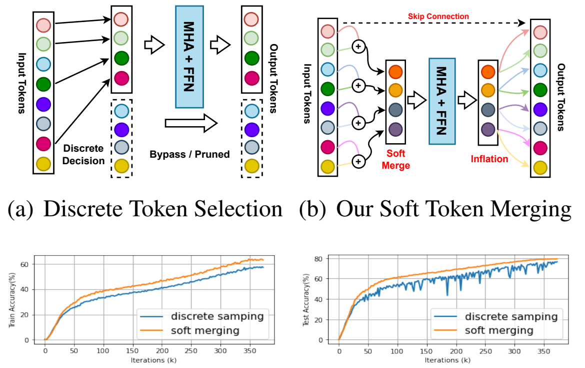
Efficient Transformer Adaptation with Soft Token Merging

Anonymous ACL submission

Abstract

We develop an approach to efficiently adapt transformer layers, driven by an objective of optimization stability and broad applicability. Unlike existing methods which adopt either simple heuristics or inefficient discrete optimization methods for token sampling, we craft a lightweight soft token merging system that maintains end-to-end differentiability while maintaining good task performance. To compensate for the potential information loss, we design a novel token inflation module to maximize functionality preservation across different transformer blocks. Experimental results across vision-only, language-only, and vision-language tasks show that our method achieves comparable accuracies while saving considerable computation costs for both training and inference. We demonstrate that these gains translate into real wall-clock speedups.



(a) Discrete Token Selection (b) Our Soft Token Merging

(c) Discrete vs. Soft: Train Accuracy w.r.t Iterations (d) Discrete vs. Soft: Test Accuracy w.r.t Iterations

Figure 1: Transformer adaption with soft token merging strategies. Different from (a) which relies on discrete token selection strategy, our soft merging scheme (b) aggregates the tokens efficiently while maintaining end-to-end differentiability. Consequently, ours yields not only (c) better fitting power during training but also (d) more robust generalization capability.

1 Introduction

Large-scale transformer, dramatically scaling up network size into the billions of parameter regime, has recently revolutionized natural language processing (NLP) (Vaswani et al., 2017; Devlin et al., 2019; Brown et al., 2020; Zaheer et al., 2020; Raffel et al., 2020), computer vision (CV) (Dosovitskiy et al., 2021; Touvron et al., 2021; Jiang et al., 2021) and multimodal applications (Radford et al., 2021; Kim et al., 2021; Chen et al., 2023d,c). However, the size of these models imposes prohibitive computation and memory consumption for both pretraining and downstream finetuning, hence motivates techniques that offer cheaper alternatives (Li et al., 2020; Gupta et al., 2021; Bondarenko et al., 2021; Kim and Hassan, 2020) to full-scale training and inference procedure. Exemplifying this situation, the desire to minimize compute and memory requirements has led to the development of token sparsification techniques, allowing large-scale transformer layers to skip computations while maintaining comparable task performance through token pruning (Hou et al., 2022; Kong et al., 2022; Xu et al., 2022; Yao et al., 2023; Xu et al., 2023) or merging (Ryoo et al., 2021; Bolya et al., 2022; Cao et al., 2023; Nawrot et al., 2023; Pietruszka et al., 2022).

Our approach incorporates these ideas, but extends the scope of applicability to various transformer-based architectures in both CV, NLP and multimodal tasks, within the context of pretraining, fully finetuning and parameter-efficient adaptation. Rather than making a discrete decision as to which token to bypass transformer layers, we propose the idea of soft token merging. Our contribution is to do so in a manner that tokens are merged while maintaining the end-to-end differentiability, saving compute by leveraging intermediate slim tokens processed by the transformer blocks without any architectural modification.

As a common practice, token reduction yields a quadratic overall efficiency improvement w.r.t token length, than training a transformer with full tokens. The general design of transformer layers suggests possible compatibility between the tokenized representations and architectural configuration, i.e. trainable weight parameters are invariant with the token length. This facilitates the desire to maintain sparsified tokens and unchanged transformer architectures. Competing recent efforts, draw inspiration from the observation that a subset of tokens may suffice the discriminative or generation tasks, In particular, token dropping (Hou et al., 2022; Yao et al., 2023; Xu et al., 2023) splits the computation from an intermediate layer and then aggregates the full-length token in the top layer to save computation. DyViT (Rao et al., 2021) adopts an attention masking strategy and auxiliary discrete optimization strategy (e.g. gumbel softmax tricks (Jang et al., 2016)) to differentiably prune tokens progressively. Kong et al. (2022); Xu et al. (2022) follows a similar strategy, adopting the masking strategy during training, which may not yield practical acceleration during training. The above discrete selection strategy, shown in Figure 1(a) is a common paradigm for most existing methods. Furthermore, these progressive token pruning methods are designed based on the nature of redundancy of visual tokens in ViT architectures, which may not directly apply to general transformer blocks for generation tasks. (e.g. machine translation).

In this paper, we develop a token merging framework around the principles of efficient optimization, offering end-to-end differentiability and maximum information preservation. Figure 1(b) illustrates key differences with prior work. Our core contributions are:

- **Efficient Soft Token Merging:** We propose a merging scheme accounting for the tokens aggregation based on the attentive information provided by themselves. This auxiliary system is computationally invariant to token length and can quickly adapt to long sequence tasks.
- **Inflation with Information Preservation:** The full token length is recovered through an inflation module, to preserve the information across different transformer blocks without affecting efficiency.
- **Better Performance and Broad Applicability:** Our method not only saves the compute but also yields excellent generalization accuracy, with the flexibility in choosing different trade-offs between efficiency and accuracy. Furthermore, adopting a merging scheme instead of masking strategy provides acceleration in terms of wall-clock training time. We demonstrate results on image classification, machine translation and visual question answering tasks, across a diverse set of transformer architectures.

2 Related Work

Token Pruning Given the property of transformers in processing arbitrary token length, several token pruning methods (Rao et al., 2021; Kong et al., 2022; Xu et al., 2022; Liang et al., 2022; Xu et al., 2023) have been proposed to progressively reducing the number of tokens for efficient inference. For example, DyViT (Rao et al., 2021) proposes a MLP predictor to dynamically sample tokens, which is trained with continuous relaxation (Jang et al., 2016) and knowledge distillation (Hinton et al., 2015). IdleViT (Xu et al., 2023) selects a subset of the image tokens in computations while bypassing the rest of tokens. These approaches are dynamic which does not directly support batching for efficient implementation. As such, a masking scheme is adopted which impairs training efficiency. However, our unique design that facilitates hardware-friendly implementation and broad application distinguishes our approach from these works. More importantly, our approach demonstrates an elegant optimization scheme with end-to-end differentiability, merely trained with task loss.

Token Merging Some other works (Ryoo et al., 2021; Bolya et al., 2022; Cao et al., 2023; Nawrot et al., 2023; Pietruszka et al., 2022) instead focus on merging tokens for efficient transformers. TokenLearner (Ryoo et al., 2021) adopts an MLP to mine important tokens in visual data hence reducing the number of tokens. ToMe (Bolya et al., 2022) reduces the number of tokens in a transformer gradually by partitioning and merging tokens in each block. PuMer (Cao et al., 2023) combines token pruning and merging works into a token re-duction framework suitable for Vision-Language models. Token pooling approaches (Nawrot et al., 2023; Pietruszka et al., 2022) average the encoded representations for efficient self-attention computation. Although token merging methods and our algorithm share the same spirit of generating efficient transformers through merging, ours gains applicability and performance with the dedicated design choice and optimization strategy.

Parameter-Efficient Fine-Tuning Parameter-Efficient Fine-Tuning (PEFT) (Houlsby et al., 2019;

Hu et al., 2022; Tang et al., 2023; Chen et al., 2023b; Yang et al., 2023; Valipour et al., 2023) adds new parameters to frozen large pre-trained LLM, enabling efficient tuning on a new training dataset. LoRA (Hu et al., 2022) is an improved PEFT method in which two matrices with lower rank are fine-tuned, approximating original matrices. This fine-tuned LoRA adapter is then used for accurate inference. Our approach not only supports fully fine-tuning but also has the flexibility in serving as an add-on to LoRA for a more paratermeter-efficient tuning scheme.

3 Method

Figure 2 illustrates the overall architecture of our system, which adapts the general transformer layer with input-dependent soft token merging and inflation with weighted replication. Given full-length tokens, our goal is to find the best token merging rule for a pre-defined transformer-based architecture, such that a smaller number of tokens is used, without incurring a decrease in task accuracy. Treating the task of finding this rule as a search problem is intractable due to the nature of binary selection optimization. Learning a mask over the tokens also presents problems, namely the difficulty of converting this mask into binary decisions, which would require inefficient auxiliary optimization during training. We therefore leverage self-attentive methods to derive the soft token merging schemes that encourage partial token usage with minimum loss in accuracy. Towards this end, we introduce the soft token merging system (Sec. 3.1) and token inflation module (Sec. 3.1), learning to dynamically reconfigure the token processing paths in a self-conditioned manner, which is compatible with different kinds of tuning approaches (Sec. 3.3).

3.1 Soft Token Merging

Input Attentive Module We introduce an end-to-end trainable module to score the encoded representations, which only passes a reduced number of tokens to the transformer block according to the merging window size p ($p = 2$ as a motivating example in Figure 8(a)). Given an input of p tokens $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_p\} \in \mathbb{R}^{p\times d}$, we first normalize and project it with trainable transformation matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{d\times d'}$:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q, \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K \tag{1}$$

where $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{p\times d'}$ and d' is set as $d/2$ in our implementation. We calculate the score matrix $\boldsymbol{s}$ from informative $\boldsymbol{q}$ and $\boldsymbol{k}$ as

$$\boldsymbol{S} = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d'}}) \in \mathbb{R}^{p\times p} \tag{2}$$

Since $\boldsymbol{Q}$ and $\boldsymbol{K}$ encode the context information of tokens, $\boldsymbol{S}$ is input-dependent, which is a simple way to derive the importance factor for each individual token. Note that different from Rao et al. (2021) which uses an MLP module to predict the scores, the additional trainable parameters $\boldsymbol{W}_Q, \boldsymbol{W}_K$ of our input attentive module are invariant to token lengths. Such a design is parameter efficient especially when sequence length scales up, e.g. for long texts or very high-resolution images.

Token-wise Weighted Sum Given the score matrix $\boldsymbol{S}$ indicating the importance factor for each token, one may directly view it as the probability for sparse token sampling. However, this makes the problem computationally intractable due to the combinatorial nature of binary states. To make the token sampling space continuous and the optimization feasible, DyViT (Rao et al., 2021) borrow the concept of learning by continuation (Wu et al., 2019; Xie et al., 2020) and adopt the Gumbel-Softmax (Jang et al., 2016) trick. This still leads to inefficient and unstable optimization, where an additional fine-tuning stage involving knowledge distillation is designed to bridge the performance gap(Rao et al., 2021). To address this issue, we simply merge the tokens through learned weighted sum to maintain end-to-end differentiability, as depicted in Figure 8(b). We calculate the score for each candidate token as:

$$\overline{\boldsymbol{S}} = [s_1, s_2, ..., s_p] = \frac{1}{p}\sum_{i=1}^{p} \boldsymbol{S}_{i,j} \tag{3}$$

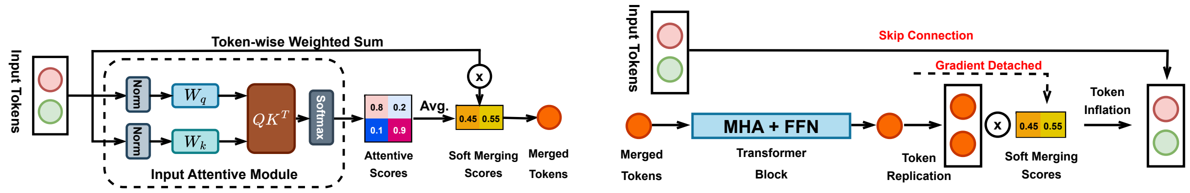
i, j denotes the index along the first (token) axis of $\boldsymbol{Q}$ and $\boldsymbol{K}$, respectively. We then obtain the merged token as:

$$\boldsymbol{x}' = \frac{1}{p}\sum_{j=1}^{p} s_j \boldsymbol{x}_j \tag{4}$$

$\boldsymbol{x}'$ is fed into the transformer block to achieve quadratic computational efficiency in terms of both time and memory:

$$\boldsymbol{y} = \text{FFN}(\text{MHA}(\boldsymbol{x}')) \tag{5}$$

where FFN and MHA denote feed-forward networks and multi-head attention in a transformer block, respectively.



(a) Input-attentive Soft Token Merging (b) Token Inflation with Replication

Figure 2: System overview. The proposed framework consists of two components (a): input-attentive soft token merging and (b): token inflation with replication. The input-attentive module is designed to build up data-dependent score matrices from input tokens of each transformer layer, serving as the importance factors for merging individual tokens through weighted sum. Merged tokens are then fed through (pretrained) transformer layer (multi-head attention + feed forward networks) with reduced computational complexity. The processed tokens are then inflated to original length through replication and rescaling for information preservation across different transformer blocks. All modules are end-to-end trainable, which are optimized by the task loss.

3.2 Inflation with Weighted Replication

Our goal is to efficiently adapt transformer architecture for various tasks. For a discriminative task (e.g. ViT for image classification) where only a single token is used in cross-entropy loss, tokens can be eliminated at certain blocks and never get sampled. However for generation tasks (e.g. encoder-decoder architecture for machine translation), it is crucial to maintain the token length during the interaction with the cross-attention layer of the decoder. To achieve general applicability, we propose a simple yet effective inflation scheme with weighted token replication. With computational cost savings already obtained, it's free to first clone the replicate $\boldsymbol{y}$ to $\boldsymbol{y}'$ with the original length. We then re-use the soft merging scores $\overline{\boldsymbol{S}}$ with gradient detached to construct the inflated tokens $\hat{\boldsymbol{y}}$:

$$\hat{\boldsymbol{y}} = \boldsymbol{X} + \boldsymbol{y}' \odot \text{detached}(\overline{\boldsymbol{S}}) \tag{6}$$

where $\odot$ is the Hadamard product and $\boldsymbol{x}$ is used in skip connection for maximum information preservation. Note that in practice detaching the gradients of $\boldsymbol{S}$ is crucial for the optimization stability, we provide detailed justification in the experimental section. Alg. 1 summarizes our soft token merging system.

3.3 Optimization

All the proposed modules can be trained in an end-to-end manner with only a task loss function. We provide three different tuning modes to accommodate various transformer applications: (1) Training the model from randomly initialized weights, (2) Given a pre-trained transformer model, we inject our token merging system without any architectural change due to the token length invariant property, and (3) One also has the flexibility to incorporate LoRA for more parameter-efficient tuning.

Algorithm 1 : Soft Token Merging

Input: Full-length tokens $\boldsymbol{x}$.
Output: Trained model $\boldsymbol{\theta}$
Initialize: Model weights $\boldsymbol{\theta}$, depth L.
for $l = 1$ **to** L **do**
  Merge $\boldsymbol{X}$ into $\boldsymbol{x}'$ using Eq. 1- 4.
  Process merged $\boldsymbol{x}'$ to $\boldsymbol{y}$ using Eq. 5.
  Inflate $\boldsymbol{y}'$ to $\hat{\boldsymbol{y}}$ using Eq. 6.
  Assign $\boldsymbol{X} = \hat{\boldsymbol{y}}$ for next layer.
end for
Back-propagate with task loss and update $\boldsymbol{\theta}$.

4 Experiments

We evaluate our approach on image-only, language-only and vision-language tasks with variants of transformer architectures. Specifically, we conduct both pretraining and evaluation on ImageNet-1K (Deng et al., 2009) for image classification, finetuning on `wmt_t2t_ende_v003` from seqio [1] for machine translation, and finetuning on VQAv2 (Goyal et al., 2017) and STVQA (Biten et al., 2019) for visual question answering.

Implementation Details For ImageNet-1K image classification, we validate our approach on the ViT-S/16 variant (Dosovitskiy et al., 2021) and follows the settings (Beyer et al., 2022) which yields significantly better performance: We use global average-pooling (GAP) instead of a class token. We adopt the learned position embeddings instead of fixed 2D sin-cos ones. We also intro-

[1] https://github.com/google/seqio

duce RandAugment (Cubuk et al., 2020) (level 10) and Mixup (Zhang et al., 2018) (probability 0.2). We implement the baseline model in Jax (Bradbury et al., 2018) and train it with Adam (Kingma and Ba, 2015), an initial learning rate of 0.001, weight decay of 0.0001 for 300 epochs on TPUv3-16 node. We choose to merge every two tokens and inject the token merging system into 4-th layer to achieve a favorably good trade-off between accuracy and efficiency. To compare with different dynamic token pruning methods implemented in Pytorch (Paszke et al., 2019), we also follow the setting in (Rao et al., 2021; Xu et al., 2023) and select the DeiT-S (12 Layers) (Touvron et al., 2021) and LV-ViT-S (16 layers) (Jiang et al., 2021) as the backbones. We finetune both models for 30 epochs on 2 NVIDIA V100 GPUs.

For machine translation, we use the T5X codebase[2] and adopt the pre-trained small and base models on C4 (Raffel et al., 2020), denoted as `t5_small` and `t5_base` respectively. `t5_small` and `t5_base` are both encoder-decoder architectures with 8 and 12 attention blocks. We finetune each model on `wmt_t2t_ende_v003` to perform the downstream machine translation tasks. Batch size is 1500 and we use 4000 warm up iterations. For each model, we use a maximum sequence length of 256 and a batch size of 128 sequences. We train with Adafactor (Shazeer and Stern, 2018) for 20k iterations, a base learning rate of 0.001 and warmup steps of 1,000 on TPUv3-16 node.

For VQA tasks, we train the recently proposed PaLI-5B model (Chen et al., 2023c) (detailed in Appendix section A.3) on VQA tasks under both fully fine-tuning and LoRA tuning settings. The image resolution is 812×812 with a patch size of 14×14, resulting in 3364 visual tokens. We apply our token merging on visual tokens output from the pre-trained ViT and set p as 2 for all variants. For both fine-tuning settings, we use the batch size of 128 and train with Adafactor for 500k iterations on TPUv3-16 node. The dropout rate is set as 0.1. For fully fine-tuning, the initial learning is $1e^{-4}$ while for LoRA with rank of 16, it's $3e^{-5}$. We also evaluate our approach in a lightweight vision-language model ViLT (0.11B, 12 Layers) (Kim et al., 2021). We implement our method in Pytorch, follow the setting in PuMer (Cao et al., 2023) to compare with DyViT (Rao et al., 2021), ToMe (Bolya et al., 2022) and PuMer (Cao et al., 2023). For a fair comparison, we adapt different configurations of merging position l to generate our model with similar FLOPs with all competitors and evaluate the accuracy/throughput trade-off on a single NVIDIA 1080Ti GPU.

[2] https://github.com/google-research/t5x

Table 1: Comparison with DyViT* on ImageNet for ViT-S/16 training from scratch over 5 random seeds.

Method	Top-1 Acc(%)	Params(M)	FLOPs(G)
Original	80.1±0.24	23.8	4.6
DyViT*	76.4±0.31	30.9	6.1
Ours	**79.3±0.18**	**24.0**	**2.9**

Table 2: Comparisons on ImageNet for fine-tuning DeiT-S. For each competing algorithm, the table reports Top-1 accuracy (%), FLOPs and inference throughput (imgs/s) from respective papers. We run our method over 5 random seeds.

Method	Top-1 Acc(%)	FLOPs(G)	Infer Tput.(imgs/s)
Original	79.8(-0.0)	4.6	2477
IdleViT	79.0(-0.8)	2.4	4072
DyViT	77.5(-2.3)	2.2	**5147**
EViT	78.5(-1.3)	2.3	3383
Evo-ViT	77.7(-2.1)	2.4	3173
ATS	78.2(-1.6)	2.3	2352
Ours	**79.3±0.1 (-0.5)**	**2.3**	4566

4.1 Results on ImageNet-1K Classification

ViT-S/16 Table 1 shows results in terms of test accuracy, trainable parameters, and training cost calculated based on overall FLOPs. We compare with a variant of DyViT (Rao et al., 2021), which is *trained from scratch* for 300 epochs. Note that additional trainable parameters of MLP prediction module and computational training overhead of masking implementation are counted. Ours achieves better test accuracy than DyViT, which suggests our soft merging method benefits the optimization process and yields better generalization performance than gumbel-softmax for sampling. Moreover, our input attentive module is lightweight and token length-invariant, which only introduces negligible parameters (0.2M) while the MLP prediction module in DyViT is 7.1M. The masking scheme in DyViT does not eliminate tokens during *training*, which yields more computational costs than training a ViT-S/16 with full-length tokens.

DeiT-S We also compare our approach with ATS (Fayyaz et al., 2022), Evo-ViT (Xu et al., 2022), EViT (Liang et al., 2022), DyViT (Rao et al., 2021) and IdleViT (Xu et al., 2023) on DeiT-S *fine-tuning*. We set the token-kept ratio $k \in$

$[0.8, 0.7, 0.6, 0.5]$ to generate different model configurations as in the respective papers. For our approach, we inject soft merging into $l \in [7, 6, 5, 4]$-th transformer block to obtain similar FLOPs as the above competitors. Results in Table 2 show that ours ($l = 4$) achieves not only better test accuracy but also faster inference throughput than those competitors ($k = 0.5$). This suggests that even without auxiliary knowledge distillation loss, our soft token merging provides more generalization capability during optimization than merely dropping the tokens. Figure 6 shows that ours yields the best accuracy and efficiency trade-offs across all configurations. Our method ($l = 4$) achieves better performance than the original DeiT-S while saving 24% FLOPs, suggesting that token merging might have an additional regularizing effect during fine-tuning. We also report more comparisons in terms of accuracy and throughput in Appendix section A.1 across different model configurations.

LV-ViT-S For LV-ViT-S *fine-tuning*, we compare our method with DyViT and IdleViT. Figure 4 shows a similar trend that ours bests accuracy-FLOPs trade-off. Appendix Table A.2 details the numbers under different model configurations.

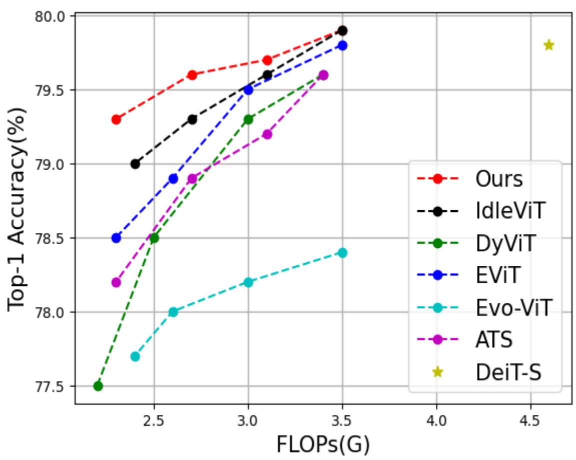


Figure 3: ImageNet-1K Top-1 accuracy-FLOPs trade-off comparison on DeiT-S fine-tuning. Ours consistently perform better than all ViT token pruning competitors.

4.2 Results on Machine Translation

We validate our approach on WMT machine translation task. Applying ViT token competitors to the encoder-decoder transformer architecture is nontrivial due to their domain-specific design of discrete optimization. As such, we only design variants of our method for self-comparison. As

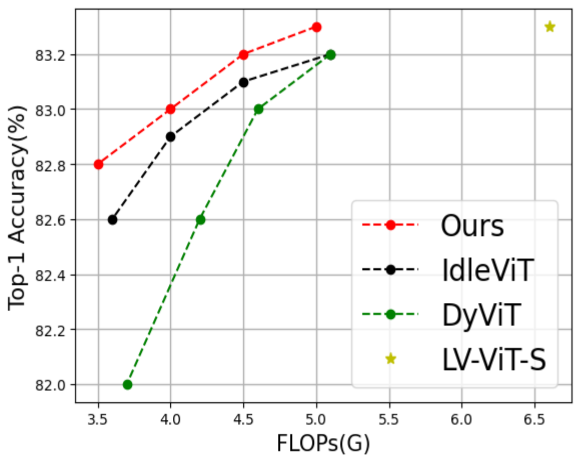


Figure 4: ImageNet-1K Top-1 accuracy-FLOPs trade-off comparison on LV-ViT-S fine-tuning. Ours consistently perform better than all competitors.

Table 3: Results of t5-small and t5-base on WMT machine translation task. The table reports BLEU score (%), training, and inference FLOPs (G) for both variants of our approach w/wo token inflation.

Method	BLEU (%)	Train/Infer FLOPs(G)
T5-small	22.9±0.27	134.9 / 3.3
Ours-w-inflat	**21.6±0.21**	121.9 / 3.1
Ours-wo-inflat	19.1±0.12	**115.0 / 3.0**
T5-base	24.3±0.29	417.1 / 51.7
Ours-w-inflat	**23.6±0.22**	355.4 / 49.4
Ours-wo-inflat	21.2±0.19	**331.6 / 46.9**

shown in Table 3, our method generalizes well to the encoder-decoder transformers T5-small and T5-base. We also validate that inflating tokens drastically improves BLEU at a reasonable cost during training and inference. This suggests that information preservation is a necessity for language generation when encoded representations interact with the target tokens in cross-attention layers.

4.3 Results on Visual Question Answering

We demonstrate the applicability of our approach to the multimodal application, visual question answering (VQA). We choose the backbone architecture of PaLI-5B, and fine-tune on VQAv2 and STVQA datasets. Since the resolution of the input image is 812×812, PaLI-5B takes the visual tokens scaling up to $3,364$. We merge the encoded tokens from a frozen pre-trained ViT without inflation since we only need the high-level visual concepts in this language generation task. The results in Table 4 show that in the context of fully fine-tuning, our approach achieves comparable accuracies while maintaining

Table 4: Results of PaLI-5B fully fine-tuning and LoRA on VQAv2 and STVQA. We report accuracy (%), training (sequences/s), and inference (tokens/s) throughputs.

Method	Accuracy (%) ↑	Train/Infer Tput. ↑
	Dataset: VQAv2	
Full-ft.	81.7±0.20	72.0 / 154.7
Ours-Full-ft	81.4±0.17	108.5 / 180.3
LoRA	79.9±0.21	74.0 / 154.6
Ours-LoRA	79.9±0.18	115.4 / 179.1
	Dataset: STVQA	
Full-ft.	77.5±0.28	67.3 / 128.5
Ours-Full-ft	76.6±0.21	99.3 / 144.7
LoRA	77.8±0.18	69.3 / 128.5
Ours-LoRA	77.3±0.16	105.3 / 144.1

Table 5: Results of ViLT on VQAv2. The table reports accuracy (%), inference throughput acceleration (×) from respective papers. We run our method over 3 random seeds.

Method	Accuracy (%) ↑	Infer Tput. ↑
Original.	69.5	1×
DyViT	67.9	1.75×
ToMe	68.4	1.79×
PuMer	68.9	1.76×
Ours	**69.1±0.1**	1.76×

a wall-clock acceleration. LoRA, as a parameter-efficient tuning approach, accelerates the training a bit without improving the inference speed. Incorporating LoRA, ours not only drastically saves training costs but also speeds up inference while maintaining comparable accuracies.

We also evaluate our approach by training another VL model ViLT. Following the settings in PuMer, we configure all methods with similar speedup and compare the accuracy over 3 runs. As shown in Table 5, our approach outperforms these competitors, which demonstrates the effectiveness of our design choices.

4.4 Analysis

Abalation Study We show the effects of turning off each of our modifications to our full optimization process (1) Full method described in Alg. 1. (2) wo-inflat.: we don't apply inflation to merged tokens. (3) wo-detach: we don't detach the gradients of the score matrix in Eq. 6. We conduct experiments using both ViT-S/16 on ImageNet and T5-small on WMT. As shown in Table 6, removing token inflation can improve the performance of ViT-S/16 by providing a subset of tokens encoded with high-abstraction visual concepts in the discriminative task. Detaching gradients of the score matrix is a necessity in stabilizing the optimization process for both architectures. We also see that both inflation and gradient detach are designed and woven to accomplish the empirical leap in the language generation task. In Figure 5(b) and 5(c), red curve and yellow curve also demonstrate that token inflation consistently improves BLEU score for both t5-small and t5-base across different model FLOPs.

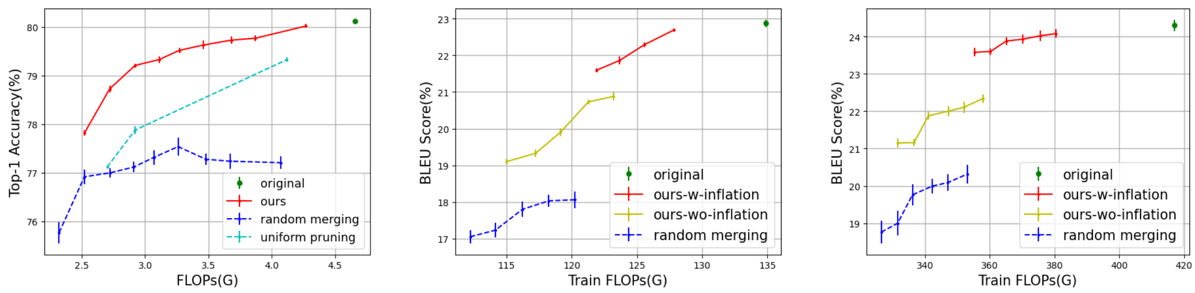
Comparison with Random Baseline In Figure 5(a), for ViT-S/16 on ImageNet-1K, we compare models obtained by (1) *uniform pruning*: a naive predefined pruning method that prunes the same percentage of dimension d in each layer, (2) *ours*: variants of our method by setting different merging positions l, and our method outperforms uniform pruning, demonstrating that token merging maintains higher generalization capacity than architectural pruning. In addition to the *uniform pruning* baseline, we also compare with a random merging baseline to further separate the contribution of the intrinsic property of token sparsification and soft merging method. Specifically, this random baseline replaces the procedure for merging entries of $\boldsymbol{S}$ in Eq. 4. Instead of using merging scores derived from the learned $\boldsymbol{S}$, it samples randomly from a uniform distribution and then normalizes the sum to 1. As shown in Figure 5 (*random merging*), *ours* consistently performs much better than this random baseline. These results, as well as the more sophisticated baselines in *uniform pruning*, demonstrate the effectiveness of our approach.

Table 6: Ablation study on inflation and gradient detach components on ImageNet-1K and WMT.

Variant	ViT-S/16 (%) ↑	T5-small (%) ↑
Full	78.4±0.15 (+0.0)	**22.9±0.27** (+0.0)
wo-inflat.	**79.3±0.18** (+0.9)	21.6±0.21 (-1.3)
wo-detach	75.3±0.10 (-3.1)	13.2±0.10 (-9.7)

Table 7: Ours still yields reasonable performance for both vision and language tasks with merging window size p enlarged to 4.

Method	ViT-S/16		T5-small	
	Train FLOPs(G) ↓	Test Acc.(%) ↑	Train FLOPs(G) ↓	BLEU (%) ↑
Original	4.6	80.1±0.24	134.9	22.9
Rand. ($p = 2$)	2.8	77.1±0.24	120.2	18.1
Rand. ($p = 4$)	1.9	76.0±0.28	115.4	15.7
Ours ($p = 2$)	2.9	79.3±0.18	121.9	21.6
Ours ($p = 4$)	2.0	78.1±0.12	117.0	19.3



(a) ViT-S/16 (b) T5-small (c) T5-base

Figure 5: Performance/FLOPs trade-offs for different variants of ViT-S/16, T5-small, and T5-base architectures. We report the results of all variants over 5 random seeds.

Table 8: Comparison with trainable token pooling. Ours has best performance consistently.

Method	ViT-S/16 Train FLOPs(G) ↓	ViT-S/16 Test Acc (%) ↑
Original	4.6	80.1±0.24
Trainable Pooling ($l = 4$)	2.9	78.0±0.16
Ours ($l = 4$)	2.9	**79.3±0.18**
Trainable Pooling ($l = 6$)	3.2	78.8±0.14
Ours ($l = 6$)	3.3	**79.7±0.14**

Investigation on Merging/Inflation Position Different from dynamic token pruning approaches which set token-kept ratios k for different model configurations, our approach realizes the flexibility by injecting merging and inflation modules at different layer positions l. Appendix section A.4 illustrates this strategy. Figure 5 investigates the performance-FLOPs trade-off curves of different variants by alternating l. Our approach not only bests accuracy among all baselines, but also appears to be more robust over different FLOPs.

Investigation on Merging Window Size The design of merging window size p gains the flexibility to explore more trade-offs between training budgets and test performance. Appendix section A.5 illustrates this strategy. Table 7 show the results for ours and random baselines, each generates trade-offs between train costs and test accuracy by alternating the window sizes ($p \in \{2, 4\}$). Ours consistently outperforms random baselines. Even with a large window size $p = 4$, ours still yields reasonable accuracy, demonstrating that the regularization effect of ours benefits generalization performance.

Connection with Trainable Pooling (Pietruszka et al., 2022) proposes an attention sparsification approach by learning to select the most informative token representations, focusing on long document summarization task, denoted as trainable pooling. Both introduce elegant optimization schemes with end-to-end differentiability, guided by merely task losses. However, ours explicitly learns self-attentive scores for token reduction without any modification to the pre-defined transformer layers (attention mechanism, architectural configuration). We generalize (Pietruszka et al., 2022) to ViT-S/16 on ImageNet-1K classification by adopting cross-attention for trainable visual token pooling at $l \in \{4, 6\}$. As shown in Table 8, ours consistently yields better performance.

5 Conclusion

We tackle a set of optimization challenges in token merging and invent a corresponding set of techniques, including soft token merging, inflation with information preservation, and parameter-efficient tuning to address these challenges. Each of these techniques can be viewed as 'add-ons' to an original part for training transformers into a corresponding one that accounts for accuracy-efficiency trade-offs. There is a detailed analysis of these add-ons and a guiding principle governing the formulation of each computational module. Together, they accelerate training and inference without impairing model accuracy – a result that uniquely separates our approach from competitors. In light of the success of our current strategy, it is interesting to subject the proposed merging system to extremely long text or video sequence tasks as a future investigation. For example, incorporating our approach with Chen et al. (2023a) to fine-tune a pre-trained LLM with an interpolated longer context window to improve efficiency while maintaining the extreme exploration capability.

References

Lucas Beyer, Xiaohua Zhai, and Alexander Kolesnikov. 2022. Better plain vit baselines for imagenet-1k. *CoRR*, abs/2205.01580.

Ali Furkan Biten, Rubèn Tito, Andrés Mafla, Lluis Gomez, Marçal Rusiñol, C.V. Jawahar, Ernest Valveny, and Dimosthenis Karatzas. 2019. Scene text visual question answering. In *ICCV*.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2021. Understanding and overcoming the challenges of efficient transformer quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7947–7969, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. 2023. PuMer: Pruning and merging tokens for efficient vision language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12890–12903, Toronto, Canada. Association for Computational Linguistics.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation. *CoRR*, abs/2306.15595.

Weize Chen, Xu Han, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2023b. Stochastic bridges as effective regularizers for parameter-efficient tuning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10400–10420, Toronto, Canada. Association for Computational Linguistics.

Xi Chen, Xiao Wang, Lucas Beyer, Alexander Kolesnikov, Jialin Wu, Paul Voigtlaender, Basil Mustafa, Sebastian Goodman, Ibrahim Alabdulmohsin, Piotr Padlewski, et al. 2023c. Pali-3 vision language models: Smaller, faster, stronger. *arXiv preprint arXiv:2310.09199*.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish V Thapliyal, James Bradbury, Weicheng Kuo, Mojtaba Seyedhosseini, Chao Jia, Burcu Karagol Ayan, Carlos Riquelme Ruiz, Andreas Peter Steiner, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. 2023d. PaLI: A jointly-scaled multilingual language-image model. In *ICLR*.

Ekin Dogus Cubuk, Barret Zoph, Jonathon Shlens, and Quoc Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

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. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*.

Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. 2022. Adaptive token sampling for efficient vision transformers. In *ECCV*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*.

Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. 2021. Memory-efficient transformers via top-k attention. In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pages 39–52, Virtual. Association for Computational Linguistics.

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

Le Hou, Richard Yuanzhe Pang, Tianyi Zhou, Yuexin Wu, Xinying Song, Xiaodan Song, and Denny Zhou. 2022. Token dropping for efficient BERT pretraining. In *Proceedings of the 60th Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), pages 3774–3784, Dublin, Ireland. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *ICML*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *ICLR*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. 2021. All tokens matter: Token labeling for training better vision transformers.

Wonjae Kim, Bokyung Son, and Ildoo Kim. 2021. Vilt: Vision-and-language transformer without convolution or region supervision. In *ICML*.

Young Jin Kim and Hany Hassan. 2020. FastFormers: Highly efficient transformer models for natural language understanding. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 149–158, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, Minghai Qin, and Yanzhi Wang. 2022. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *ECCV*.

Bingbing Li, Zhenglun Kong, Tianyun Zhang, Ji Li, Zhengang Li, Hang Liu, and Caiwen Ding. 2020. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3187–3199, Online. Association for Computational Linguistics.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2022. Evit: Expediting vision transformers via token reorganizations. In *ICLR*.

Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. 2023. Efficient transformers with dynamic token pooling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6403–6417, Toronto, Canada. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Michał Pietruszka, Łukasz Borchmann, and Łukasz Garncarek. 2022. Sparsifying transformer models with trainable representation pooling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8616–8633, Dublin, Ireland. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*.

Michael S. Ryoo, A. J. Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. 2021. Tokenlearner: Adaptive space-time tokenization for videos. In *NeurIPS*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*.

Moming Tang, Chengyu Wang, Jianing Wang, Chuanqi Tan, Songfang Huang, Cen Chen, and Weining Qian. 2023. XtremeCLIP: Extremely parameter-efficient tuning for low-resource vision language understanding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6368–6376, Toronto, Canada. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. UL2: unifying language learning paradigms. In *ICLR*.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *ICML*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. DyLoRA:

Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287, Dubrovnik, Croatia. Association for Computational Linguistics.

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

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*.

Zhenda Xie, Zheng Zhang, Xizhou Zhu, Gao Huang, and Stephen Lin. 2020. Spatially adaptive inference with stochastic feature sampling and interpolation. In *ECCV*.

Xuwei Xu, Changlin Li, Yudong Chen, Xiaojun Chang, Jiajun Liu, and Sen Wang. 2023. No token left behind: Efficient vision transformer via dynamic token idling. *arXiv preprint arXiv:2310.05654*.

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

Xiaocong Yang, James Y. Huang, Wenxuan Zhou, and Muhao Chen. 2023. Parameter-efficient tuning with special token adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 865–872, Dubrovnik, Croatia. Association for Computational Linguistics.

Zhewei Yao, Xiaoxia Wu, Conglong Li, Connor Holmes, Minjia Zhang, Cheng Li, and Yuxiong He. 2023. Efficient large-scale transformer training via random and layerwise token dropping.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid loss for language image pre-training. *CoRR*, abs/2303.15343.

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

A Appendix

We further compare ours and recently proposed progressive token pruning approaches on Deit-S by showing additional Top-1 accuracy on ImageNet-1K, FLOPs, and inference throughput. Table 9, 10, 11 and 12 demonstrate that our approach outperforms all the competitors consistently.

Table 9: Comparisons on ImageNet for fine-tuning DeiT-S. For competing methods, we set the token kept ratio as 0.4 while for our approach the merging position l are set as 3.

Method	Top-1 Acc(%)	FLOPs(G)	Infer Tput.(imgs/s)
IdleViT	78.4	2.1	4363
DyViT	76.0	1.9	**5741**
EViT	77.6	2.0	3717
Evo-ViT	77.5	2.1	3548
ATS	76.4	2.0	2580
Ours	**78.7**	2.0	4843

A.1 More results in DeiT-S

Table 10: Comparisons on ImageNet for fine-tuning DeiT-S. For competing methods, we set the token kept ratio as 0.6 while for our approach the merging position l are set as 5.

Method	Top-1 Acc(%)	FLOPs(G)	Infer Tput.(imgs/s)
IdleViT	79.3	2.7	3693
DyViT	78.5	2.5	**4474**
EViT	78.9	2.6	3045
Evo-ViT	78.0	2.6	2998
ATS	78.9	2.7	2229
Ours	**79.6**	2.7	4002

Table 11: Comparisons on ImageNet for fine-tuning DeiT-S. For competing methods, we set the token kept ratio as 0.7 while for our approach the merging position l are set as 6.

Method	Top-1 Acc(%)	FLOPs(G)	Infer Tput.(imgs/s)
IdleViT	79.6	3.1	3361
DyViT	79.3	3.0	3390
EViT	79.5	3.0	2621
Evo-ViT	78.2	3.0	2606
ATS	79.2	3.1	2161
Ours	**79.7**	3.1	**3408**

Table 12: Comparisons on ImageNet for fine-tuning DeiT-S. For competing methods, we set the token kept ratio as 0.8 while for our approach the merging position l are set as 7.

Method	Top-1 Acc(%)	FLOPs(G)	Infer Tput.(imgs/s)
IdleViT	**79.9**	3.5	3031
DyViT	79.6	3.4	**3405**
EViT	79.8	3.5	2286
Evo-ViT	78.4	3.5	2293
ATS	79.6	3.4	2036
Ours	**79.9**	3.5	3321

A.2 More results in LV-ViT-S

We detail the number in Figure 4 in terms of Top-1 accuracy and FLOPs, as shown in Table 13, 14, 15 and 16. We additionally provide inference throughput to demonstrate the wall-clock acceleration.

Table 13: Comparisons on ImageNet for fine-tuning LV-ViT-S. For competing methods, we set the token kept ratio as 0.8 while for our approach the merging position l are set as 7.

Method	Top-1 Acc(%)	FLOPs(G)	Infer Tput.(imgs/s)
IdleViT	83.2	5.1	855
DyViT	83.2	5.1	958
Ours	**83.3**	5.0	**970**

Table 14: Comparisons on ImageNet for fine-tuning LV-ViT-S. For competing methods, we set the token kept ratio as 0.7 while for our approach the merging position l are set as 6.

Method	Top-1 Acc(%)	FLOPs(G)	Infer Tput.(imgs/s)
IdleViT	83.1	4.5	938
DyViT	83.0	4.6	**1077**
Ours	**83.2**	4.5	1002

A.3 Details of PaLI-5B (Chen et al., 2023c)

Different from ViLT (Kim et al., 2021) which jointly pass the linear projected image patches and text tokens to a multimodal transformer architecture, PaLI-5B first encodes the image into visual tokens with 2B SigLIP ViT (contrastively pretrained parameters) (Zhai et al., 2023) and passes the visual tokens together with text query tokens to a 3B encoder-decoder UL2 transformer (Tay et al., 2023) that generates a text output. In the experiments, we use 812×812 image resolution to demonstrate the efficiency and effectiveness of our token merging approach.

A.4 Illustration of Merging Position l

Different from existing progressive token pruning works, our system facilitates different trade-off con-

Table 15: Comparisons on ImageNet for fine-tuning LV-ViT-S. For competing methods, we set the token kept ratio as 0.6 while for our approach the merging position l are set as 5.

Method	Top-1 Acc(%)	FLOPs(G)	Infer Tput.(imgs/s)
IdleViT	82.9	4.0	1040
DyViT	82.6	4.2	**1206**
Ours	**83.0**	4.0	1188

Table 16: Comparisons on ImageNet for fine-tuning DeiT-S. For competing methods, we set the token kept ratio as 0.5 while for our approach the merging position l are set as 4.

Method	Top-1 Acc(%)	FLOPs(G)	Infer Tput.(imgs/s)
IdleViT	82.6	3.6	1131
DyViT	82.0	3.7	1321
Ours	**82.8**	3.5	**1378**

figurations via injecting the merging module to a different transformer block, depicted as merging position l in Figure 7. The merging position l effectively adapts the portion of transformer blocks that take reduced tokens, hence realizing different efficiency and accuracy trade-offs.

A.5 Illustration of Merging Window Size p

As shown in Figure 8, we illustrate the merging score matrices with different window size. Our approach has the flexibility in aggregating p local tokens into one with self-attentive importance scores, which is beneficial in maintaining reasonable task performance even with a large p.

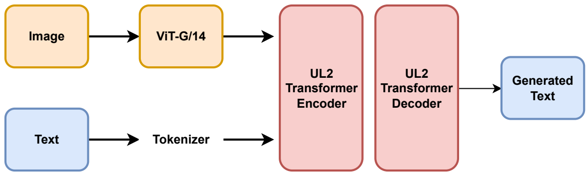


Figure 6: Overview of PaLI-5B.

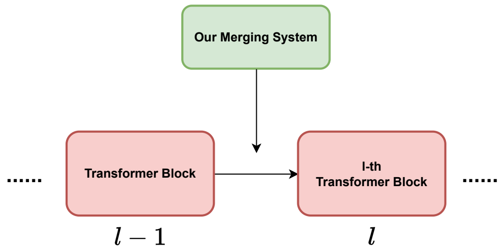
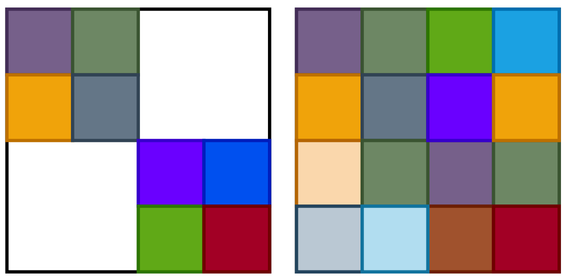


Figure 7: Illustration of applying our merging system to position l.



(a) Merge $p = 2$ tokens to 1 (b) Merge $p = 4$ tokens to 1

Figure 8: Illustration of different merging window sizes.