Efficient Transformer Adaptation with Soft Token Merging

Xin Yuan\textsuperscript{1,*}, Hongliang Fei\textsuperscript{2}, Jinoo Baek\textsuperscript{2}
\textsuperscript{1}University of Chicago \textsuperscript{2}Google
\{yuanx@uchicago.edu, hongliangfei, jinoo\}@google.com

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.

1. Introduction

Large-scale transformer, dramatically scaling up network size into the billions of parameter regime, has recently revolutionized natural language processing (NLP) [6, 14, 35, 43, 50], computer vision (CV) [15, 24, 41] and multimodal applications [10, 11, 25, 34]. 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 [4, 18, 26, 29] 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 [20, 28, 46, 47, 49] or merging [3, 7, 31, 33, 37].

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 pre-training, 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 [20, 46, 49] splits the computation from an intermediate layer and then aggregates the full-length token in the top layer to save computation. DyViT [36] adopts an attention masking strategy and auxiliary discrete optimiza-
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 [28, 30, 36, 46, 47] have been proposed to progressively reducing the number of tokens for efficient inference. For example, DyViT [36] proposes a MLP predictor to dynamically sample tokens, which is trained with continuous relaxation [23] and knowledge distillation [19]. Idle-ViT [46] 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 [3, 7, 31, 33, 37] instead focus on merging tokens for efficient transformers. TokenLearner [37] adopts an MLP to mine important tokens in visual data hence reducing the number of tokens. ToMe [3] reduces the number of tokens in a transformer gradually by partitioning and merging tokens in each block. PuMer [7] combines token pruning and merging works into a token re-duction framework suitable for Vision-Language models. Token pooling approaches [31, 33] 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) [9, 21, 22, 39, 42, 48] adds new parameters to frozen large pre-trained LLM, enabling efficient tuning on a new training dataset. LoRA [22] 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 parameter-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 8a). Given an input of $p$ tokens $X = \{x_1, x_2, \ldots, x_p\} \in \mathbb{R}^{p \times d}$, we first normalize and project it with trainable transformation matrices $W_Q, W_K \in \mathbb{R}^{d \times d'}$:

$$Q = XW_Q, K = XW_K$$

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

$$S = \text{softmax}(\frac{QK^T}{\sqrt{d'}}) \in \mathbb{R}^{p \times p}$$

Since $Q$ and $K$ encode the context information of tokens, $S$ is input-dependent, which is a simple way to derive the importance factor for each individual token. Note that different from Rao et al. [36] which uses an MLP module to predict the scores, the additional trainable parameters $W_Q, W_K$ 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 $S$ 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 [36] borrow the concept of learning by continuation [44, 45] and adopt the Gumbel-Softmax [23] 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[36]. To address this issue, we simply merge the tokens through learned weighted sum to maintain end-to-end differentiability, as depicted in Figure 8b. We calculate the score for each candidate token as:

$$\mathbf{S} = [s_1, s_2, \ldots, s_p] = \frac{1}{p} \sum_{i=1}^{p} S_{i,j}$$

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

$$x' = \frac{1}{p} \sum_{j=1}^{p} s_j x_j$$

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

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

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

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 $y$ to $y'$ with the original length. We then re-use the soft merging scores $\mathbf{S}$ with gradient detached to construct the inflated tokens $\hat{y}$:

$$\hat{y} = X + y' \odot \text{detached}(\mathbf{S})$$
where \( \odot \) is the Hadamard product and \( x \) is used in skip connection for maximum information preservation. Note that in practice detaching the gradients of \( S \) is crucial for the optimization stability, we provide detailed justification in the experimental section. Alg. 1 summarizes our soft token merging system.

**Algorithm 1 : Soft Token Merging**

**Input:** Full-length tokens \( x \).

**Output:** Trained model \( \theta \)

**Initialize:** Model weights \( \theta \), depth \( L \).

for \( i = 1 \) to \( L \) do

- Merge \( X \) into \( x' \) using Eq. 1- 4.
- Process merged \( x' \) to \( y' \) using Eq. 5.
- Inflate \( y' \) to \( \hat{y} \) using Eq. 6.
- Assign \( X = \hat{y} \) for next layer.

end for

Back-propagate with task loss and update \( \theta \).

### 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.

### 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 [13] for image classification, finetuning on wmt\_t2t\_ende\_v003 from seqio\(^1\) for machine translation, and finetuning on VQA\_v2 [17] and STVQA [2] for visual question answering.

**Implementation Details** For ImageNet-1K image classification, we validate our approach on the ViT-S/16 variant [15] and follows the settings [1] 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 introduce RandAugment [12] (level 10) and Mixup [52] (probability 0.2). We implement the baseline model in Jax [5] and train it with Adam [27], 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 [32], we also follow the setting in [36, 46] and select the DeiT-S (12 Layers) [41] and LV-ViT-S (16 layers) [24] 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 [35], 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 [38] 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 [10] on VQA tasks under both fully fine-tuning and LoRA tuning settings. Different from ViLT [25] 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) [51] and passes the visual tokens together with text query tokens to a 3B encoder-decoder UL2 transformer [40] that generates a text output. The image resolution is \( 812 \times 812 \) with a patch size of \( 14 \times 14 \), resulting in 3364 visual tokens to demonstrate the efficiency and effectiveness of our token merging approach. 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 Lay-

---

\(^1\)https://github.com/google/seqio

\(^2\)https://github.com/google-research/t5x
ers) [25]. We implement our method in Pytorch, follow the setting in PuMer [7] to compare with DyViT [36], ToMe [3] and PuMer [7]. 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.

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 Acc(%)</th>
<th>Params(M)</th>
<th>FLOPs(G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>80.1±0.24</td>
<td>23.8</td>
<td>4.6</td>
</tr>
<tr>
<td>DyViT*</td>
<td>76.4±0.31</td>
<td>30.9</td>
<td>6.1</td>
</tr>
<tr>
<td>Ours</td>
<td>79.3±0.18</td>
<td>24.0</td>
<td>2.9</td>
</tr>
</tbody>
</table>

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.

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

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 [36], 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 [16], Evo-ViT [47], EViT [30], DyViT [36] and IdleViT [46] 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 4 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 regularization effect during fine-tuning. We also report more comparisons in terms of accuracy and throughput in Appendix across different model configurations.

LV-ViT-S For LV-ViT-S fine-tuning, we compare our method with DyViT and IdleViT. Figure 5 shows a similar trend that ours bests accuracy-FLOPs trade-off. Results in Table 3 show that ours ($l = 4$) achieves better test accuracy and faster inference throughput than those competitors ($k = 0.5$) simultaneously. Appendix details the numbers under different model configurations.

Table 3. Comparisons on ImageNet for fine-tuning LV-ViT-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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 Acc(%)</th>
<th>FLOPs(G)</th>
<th>Infer Tput.(imgs/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IdleViT</td>
<td>82.6</td>
<td>3.6</td>
<td>1131</td>
</tr>
<tr>
<td>DyViT</td>
<td>82.0</td>
<td>3.7</td>
<td>1321</td>
</tr>
<tr>
<td>Ours</td>
<td>82.8</td>
<td>3.5</td>
<td>1378</td>
</tr>
</tbody>
</table>

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 shown in Table 4, 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 $812 \times 812$, 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 5 show that in the context of fully fine-tuning, our approach achieves comparable accuracies while maintaining 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 6, 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-infl. : 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 7, 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 6, 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 6a, 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 model 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 $S$ in Eq. 4. Instead of using merging scores derived from the learned $S$, it samples randomly from a uniform distribution and then normalizes the sum to 1. As shown in Figure 6 (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.

**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 $l$, 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.

Figure 6 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. As shown in Figure 8, we illustrate the merging score matrices with different window size. Our approach has the flexibility to aggregate $p$ local tokens into one with self-attentive importance scores, which is beneficial in maintaining rea-
Figure 6. 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.

Figure 7. Illustration of applying merging system to position $l$.

Figure 8. Illustration of different merging window sizes.

Table 8 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.

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. [8] to fine-tune a pre-trained LLM with an interpolated longer context window to improve efficiency while maintaining the extreme exploration capability.
References


[49] Zhewei Yao, Xiaoxia Wu, Conglong Li, Connor Holmes, Minjia Zhang, Cheng Li, and Yuxiong He. Efficient large-scale transformer training via random and layerwise token dropping, 2023. 1

