This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3605239

PRANCE: Joint Token-Optimization and Structural Channel-Pruning for Adaptive ViT Inference

Ye Li*, Chen Tang*, Yuan Meng†, *Member, IEEE*, Jiajun Fan, Zenghao Chai, Xinzhu Ma, Zhi Wang†, *Senior Member, IEEE*, Wenwu Zhu†, *Fellow, IEEE*

***Abstract*—The troublesome model size and quadratic computational complexity associated with token quantity pose significant deployment challenges for Vision Transformers (ViTs) in practical applications. Despite recent advancements in model pruning and token reduction techniques speed up the inference speed of ViTs, these approaches either adopt a fixed sparsity ratio or overlook the meaningful interplay between architectural optimization and token selection. Consequently, this *static* and *single-dimension* compression often leads to pronounced accuracy degradation under aggressive compression rates, as they fail to fully explore redundancies across these two orthogonal dimensions. Therefore, we introduce PRANCE, a framework which can jointly optimize activated channels and tokens on a per-sample basis, aiming to accelerate ViTs' inference process from a unified data and architectural perspective. However, the joint framework poses challenges to both architectural and decision-making aspects. Firstly, while ViTs inherently support variable-token inference, they do not facilitate dynamic computations for variable channels. To overcome this limitation, we propose a meta-network using weight-sharing techniques to support arbitrary channels of the Multi-Head Self-Attention (MHSA) and Multi-Layer Perceptron (MLP) layers, serving as a foundational model for architectural decision-making. Secondly, simultaneously optimizing the model structure and input data constitutes a combinatorial optimization problem with an extremely large decision space, reaching up to around 10^{14}, making supervised learning infeasible. To this end, we design a lightweight selector employing Proximal Policy Optimization algorithm (PPO) for efficient decision-making. Furthermore, we introduce a novel "Result-to-Go" training mechanism that models ViTs' inference process as a Markov decision process, significantly reducing action space and mitigating delayed-reward issues during training. Additionally, our framework simultaneously supports different kinds of token optimization methods such as pruning, merging, and sequential pruning-merging strategies. Extensive experiments demonstrate the effectiveness of PRANCE in reducing FLOPs by approximately 50%, retaining only about 10% of tokens while achieving lossless Top-1 accuracy. The code is available at https://github.com/ChildTang/PRANCE.**

***Index Terms*—Vision Transformer, Token Optimization, Structure Optimization, Model Lightweight.**

I. Introduction

VISION Transformers [1] have emerged as cutting-edge architectures across various fields of machine learning, including classification [2], detection [3], [4], segmentation [5], [6], multi-modal modeling [6]–[8], *etc.* Multi-Head Self-Attention, the core of the Transformers, empowers global representation modeling by dynamically weighting each token within the input sequence. However, the quadratic increased complexity of MHSA with respect to the number of tokens, coupled with the model size, significantly exacerbates deployment challenges.

The computational overheads of ViTs are primarily concentrated in *(i)* the number of embedding dimensions C and *(ii)* the quadratic complexity with the number of tokens N. Therefore, there have been two concurrent research directions that aim to improve the efficiency of ViTs including *(i)* structure compression and *(ii)* token optimization. As one of the most direct ways, the former one leverages the conventional deep model compression techniques, *i.e.*, model pruning [9], [10], weight quantization [11]–[13], and lightweight model design [14]–[16], to remove the redundant components of ViTs. For example, NViT [17] adopts structural pruning in the fields of CNNs compression to prune the channels with Hessian information. ElasticViT [15] and AutoFormer [14] automate the design process of ViTs with the help of two-stage neural architecture search [16], [18]. On the other hand, token optimization methods work on directly manipulating the number of tokens with a *predefined* token keep ratio, which is a kind of Transformer-specific technique in contrast to model compression due to the support of variable token length in MHSA. Specifically, token optimization methods can be divided into pruning-based methods and merging-based methods. Pruning-based methods [19], [20] remove the uninformative tokens

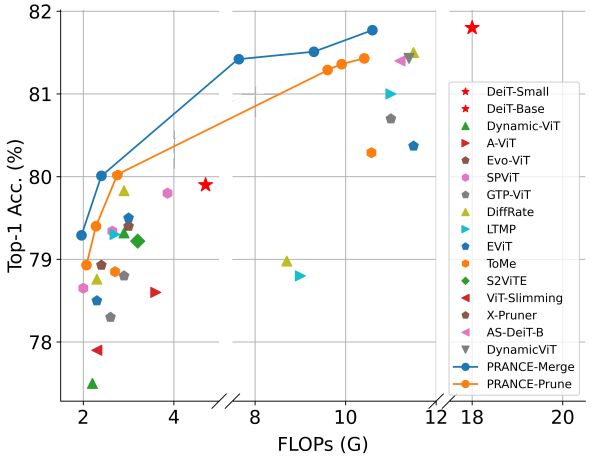


Fig. 1. **Comparison of PRANCE with SOTA methods.** PRANCE achieves both higher Top-1 accuracy and lower complexity (FLOPs) in ImageNet.

Ye Li, Chen Tang, Yuan Meng, Zenghao Chai, Zhi Wang, Wenwu Zhu are with Tsinghua University, China. Ye Li and Zhi Wang are also with Tsinghua Shenzhen International Graduate School, Tsinghua University. Xinzhu Ma is with MMLab, The Chinese University of Hong Kong, China.
Email: wangzhi@sz.tsinghua.edu.cn, {yuanmeng, wwzhu}@tsinghua.edu.cn.

*Equal contribution. †Corresponding author.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3605239

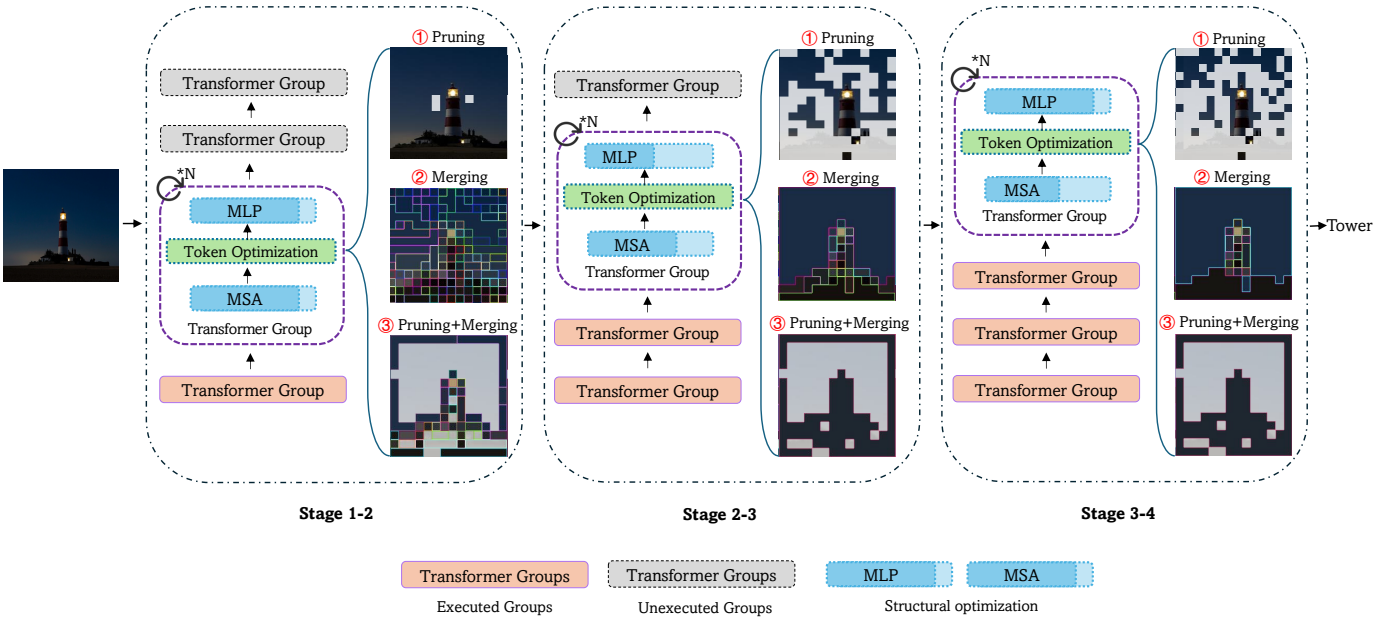


Fig. 2. **Illustration of the inference process of PRANCE.** PRANCE is a lightweight framework for ViTs that jointly optimizes model structure and data. First of all, the framework divides the ViT model into four groups according to the inference sequence, each containing multiple ViT blocks. During inference, the selector utilizes the features of each group step by step to decide the model channel dimensions and token numbers for them, aiming to minimize FLOPs while ensuring accuracy. Moreover, PRANCE supports three main token optimization methods: pruning, merging, and pruning-then-merging.

progressively during inference according to the calculated importance score. For example, DynamicViT [19] incorporates a lightweight MLP layer [21], [22] for evaluating and pruning token values. Merging-based methods [23], [24] reduce token quantity by progressively merging tokens with high similarities during inference, *e.g.,* ToMe [23] measures token correlations based on the cosine similarity of their K matrices and merges them by calculating mean values. Based on the above methods, several advanced evaluation mechanisms and fusion methods [25]–[27] have been proposed to achieve higher compression rates while maintaining accuracy.

While significant reductions in complexity (*e.g.*, FLOPs, latency, model size, *etc.*) have been achieved, the above methods still have some limitations. From the perspective of structural reduction, they typically search for several independent low-complexity models for various downstream tasks by resorting to *data-agnostic* methods, therefore remaining significant redundancies in the model structures when applied to different data samples. In other words, they ignore that samples with varying recognition difficulties often carry different amounts of useful information. For example, consider two images of the same size: one is an apple against a white wall, and the other is a vibrant cityscape at night. Clearly, the second image contains significantly more information than the first, and thus the uninformative input should be applied more simplified architecture for efficiency. That is to say, the number of channels utilized by ViTs should dynamically adjust when processing images of varying complexity. From the perspective of data optimization, as the Transformer natively supports variable input lengths, recent studies tend to progressively evict the uninformative tokens during the inference process for the inputs, and the eviction tokens will be directly removed [19] or merged into other tokens [23]. However, token optimization typically faces significant performance degradation in the high compression ratio, this makes recent studies employ rather sophisticated token matching and reduction techniques (e.g., NAS-based search [25]) to retain performance but inevitably compromise the runtime efficiency and implementation simplicity. What's more, whether optimizing the model architecture or the data, the essence lies in eliminating redundant data and preserving effective data to achieve the optimal trade-off between model accuracy and FLOPs. Therefore, they are not entirely orthogonal and cannot be simply combined. Based on these works, we would like to ask a question:

How to adaptively optimize both the model architecture and tokens for each sample simultaneously to achieve the optimal accuracy-FLOPs trade-off?

To answer this question, two challenges need to be resolved: *(i)* although ViTs inherently support a variable number of tokens, they do not support a variable dimension of channels, and *(ii)* the optimization space created by combined optimization is excessively large, making it difficult to find the optimal solution. It is non-trivial to search for an optimal compression ratio for both architectural and token-level optimization by naively combining the existing methods in these two fields, as the resulting decision space can reach up to around 10^{12}. In this paper, we propose the PRANCE framework to address the above challenges. By optimizing both the channels and tokens from the perspectives of model architecture and data, we aim to minimize FLOPs while ensuring accuracy across different samples, the overall inference process is shown in Fig. 2.

PRANCE starts with a meta-network training process to support arbitrary channels of the MHSA and MLP layer. We adopt the weight-sharing technique to allow the smaller channels to be a subset of the large channels [14], [15], then the weights of different channel candidates can be coupled

and learned together. To simulate the architectural decision made by the selector we introduced after, we perform random sampling to select different architectures at each step of meta-network training, and the resulting model could receive variable channels after convergence. It is noteworthy that the training of meta-network is only performed once and uses all tokens by default (as in conventional ViTs training), which aims to improve the training stability. Then, to solve the combinatorial optimization problem in the mixed-decision space, where the decision values are divided into channel dimension selection and token optimization ratio decisions, we consider leveraging PPO to conduct efficiency learning. By modeling the decision-making process of architecture and tokens as a Markov process, along with a newly designed "Result-to-Go" mechanism, we achieve accurate decision estimation for each action taken by the selector. Moreover, we further experiment with three primary token optimization strategies: pruning, merging, and pruning-then-merging, and show that our frameworks can seamlessly be compatible with them. Surprisingly, we have observed even the simplest token optimization strategy can surpass previous advanced methods, which further demonstrates the effectiveness of our framework. For example, PRANCE achieved Top-1 accuracies of 72.38%, 79.98%, and 81.77% on ViT-tiny, ViT-small, and ViT-base models, respectively, while requiring only 0.87 GFLOPs, 2.38 GFLOPs, and 10.59 GFLOPs and saving up to 53% FLOPs. We build up a new Pareto front of the Vision Transformer compressions, which sheds light on the importance of joint optimization for both architectural and data aspects.

To summarize, our contributions are as follows:

(1) We propose a high-accuracy, low-FLOPs framework, PRANCE, for ViTs compression, which allows sample-wise joint optimization of token numbers N and channel dimensions C during inference. This framework optimizes both model architecture and data dimensions while being compatible with pruning, merging, and pruning-then-merging token optimization methods. For samples with different complexities, PRANCE dynamically selects the channels and tokens with more information to achieve the optimal efficiency-accuracy trade-off.

(2) We construct a meta-network with variable channels. Specifically, we train a high-performing meta-network using weight-sharing techniques under multiple selectable MHSA channels and MLP expansion ratios, providing PRANCE with a foundational model capable of dynamically adjusting channel dimensions and token numbers.

(3) We propose a lightweight PPO-based selector for ViTs and introducing a new training mechanism "Result-to-Go", which significantly reduces the selection space of combinatorial optimization problems. This selector can progressively optimize the data and model structures while maintaining the compact action space dimensions.

(4) We conduct extensive experiments and demonstrate that PRANCE exhibits excellent performance. It achieves higher Top-1 accuracy with lower FLOPs across Tiny, Small, and Base scales, surpassing various state-of-the-art methods.

II. Related Work

Vision Transformers Compression. To deploy ViTs on resource-limited devices, recent advances lean upon the conventional model compression techniques to lessen the *architectural redundancies* of these over-parameterized models, including model pruning [9], [10], [17], [28], quantization [11], [12], [29], [30], and lightweight architecture design [14], [15], [31]. They explore effective solutions for lightweight ViTs from three aspects: reducing the number of model channels, lowering the precision of model storage, and finding more optimal lightweight structures. Specifically, NViT [17] constructs the final model by establishing a global importance score ranking, observing the trend of dimension changes in the pruned network structure, and reallocating parameters. FQ-ViT [29] adaptively quantizes all structures of the ViTs model while maintaining results similar to those of the full precision model. Autoformer [14] introduces the weight-sharing mechanism into the ViTs and obtains a multi-scale high-precision super network through supervised training. Then ElasticViT [15] improves the sampling strategies for the supernet training, designing a high-precision, lightweight ViTs model that supports a wide range of mobile devices with varying computational power. The above methods explore the possibilities of architectural optimization for ViTs in various domains and achieve excellent results at the task level. However, it is evident that the requirements for model parameters and quantization bit-width should differ when processing simple and complex images with ViTs. Additionally, these methods overlook optimization in the data dimension: ViTs can achieve accurate results by focusing only on the important parts of different samples, while the unimportant parts lead to redundant computations. Therefore, there is tremendous potential for optimization by jointly optimizing the structure dimension and the data dimension at the sample level.

Token Optimization for Vision Transformers. From the perspective of data optimization, reducing the less informative parts of the data can effectively lower computational complexity while maintaining model performance. Unlike CNN-based models, which extract image features through convolution, transformer-based models encode data into tokens and get the semantics of them through attention mechanism. Consequently, their computational complexity grows quadratically with the number of tokens. Therefore, extensive efforts have been made to reduce the token number, and these efforts can be divided into two primary paradigms: token pruning [19], [32]–[34] and token merging [23]–[25], [35]–[37]. E-ViT [20] believes that the importance of tokens is reflected in the `<CLS>` token. Therefore, it sorts the tokens based on the value of the `<CLS>` token, retaining the important tokens based on the manually defined rate and merging the less important ones into a single token. DynamicViT [19] removes the tokens by inserting an additional MLP layer in adjacent ViTs blocks to learn the pruning decision, and achieves sample-wise token pruning. A-ViT [33] achieves dynamic sample-wise token pruning by adopting a single neuron in the MLP layer to compute the stopping score for each token. On the other hands, the merging-based approach posits that directly

discarding tokens results in information loss. By merging similar tokens, we can also reduce the number of tokens while preserving more information. ToMe [23], [24] utilize the Bipartite Soft Matching to calculate the distance between tokens and then merge the similar tokens. Besides, recent methods typically require more complex token assessment and matching techniques. For example, Diffrate [25] utilizes pruning followed by merging to further improve compression rates while maintaining good accuracy. BAT [26] needs a matching-then-clustering strategy to identify the importance and diversity of tokens, and Zero-TPrune [27] develops a graph-based matching to calculate the similarity of tokens. However, the above methods only consider data optimization, ignoring the model structural redundancy and the coupling between data and structure. Therefore, there remains significant optimization space for ViTs.

In this paper, we have demonstrated that by using only basic pruning, merging, and combinatorial methods, complemented by joint architectural optimization, we can also achieve superior performance-efficiency trade-offs compared to previous methods. Our approach further eschews the intricate token-to-token matching mechanisms required by merging techniques, which could slow down the inference on real hardware.

III. Method

Fig. 3 illustrates the framework of the proposed PRANCE, which involves two steps: firstly, pretraining a meta-network of ViTs with variable channels through simulated channel selection decisions, secondly, segmenting every three blocks of the ViTs into distinct groups, and integrating a PPO-based lightweight selector between groups for conducting sample-wise architectural decisions and token selections, and training the selector through Reinforcement Learning (RL). After that, we also fine-tune the ViTs backbone to align the representation of ViTs and the selector for optimal performance.

A. Preliminary of Computation Complexity

The computation of the Transformer-based models mainly consists of two parts: MHSA and Feed-Forward Network (FFN). Suppose the input dimension is (N, C), where N is the token number and C is the embedding dimension of the token. Then the computational complexity of MHSA is $\mathcal{O}(4NC^2 + 2N^2C)$, the computational complexity of FFN is $\mathcal{O}(8NC^2)$, and the total computational complexity is $\mathcal{O}(12NC^2+2N^2C)$. Reducing the computational complexity of such models requires attention to two aspects: from the perspective of model architecture, we can optimize the token length C, and from the perspective of data, we can optimize the number of tokens N. Additionally, considering that the effective number of tokens and the required model architecture may vary for different samples, performing sample-wise optimization on these two aspects can ideally yield optimal results.

B. Channel-elastic Meta-network Training

To support channel selection on different-sized ViTs models (*e.g.*, ViT-tiny, ViT-small, ViT-base, etc), we train several meta ViTs with variable channel and MLP ratio by making the ViTs perceive the architectural changes. Specifically, we use a set of pre-defined embedding dimensions and MLP ratios of the model, as shown in Tab. I. To support the variable channel, we enable the MHSA layers can be assigned a specific embedding dimension ϕ:

$$\texttt{MHSA}(\mathbf{x};\phi) = \texttt{Softmax}\left(\frac{\mathbf{Q}_\phi(\mathbf{K}_\phi)^\mathsf{T}}{\sqrt{\phi}}\right)\mathbf{V}_\phi, \tag{1}$$

where $\mathbf{Q}_\phi \in \mathbb{R}^{(N+1)\times\phi}$, $\mathbf{K}_\phi \in \mathbb{R}^{(N+1)\times\phi}$ and $\mathbf{V}_\phi \in \mathbb{R}^{(N+1)\times\phi}$ are projected matrices with a given embedding dimension of ϕ and the input $\mathbf{x} \in \mathbb{R}^{(N+1)\times C_{in}}$:

$$\mathbf{Q}_\phi = \mathbf{x}(\mathbf{W}^q_\phi)^\mathsf{T} \qquad \mathbf{K}_\phi = \mathbf{x}(\mathbf{W}^k_\phi)^\mathsf{T} \qquad \mathbf{V}_\phi = \mathbf{x}(\mathbf{W}^v_\phi)^\mathsf{T}, \tag{2}$$

where the projection weights $\mathbf{W}^q_\phi \in \mathbb{R}^{\phi\times C}$, $\mathbf{W}^k_\phi \in \mathbb{R}^{\phi\times C}$, $\mathbf{W}^v_\phi \in \mathbb{R}^{\phi\times C}$ are sliced from their full weights using first ϕ channels [15], [16]:

$$\mathbf{W}^q_\phi = \mathbf{W}^q[:\phi, :C_{in}] \quad \mathbf{W}^k_\phi = \mathbf{W}^k[:\phi, :C_{in}] \quad \mathbf{W}^v_\phi = \mathbf{W}^v[:\phi, :C_{in}]. \tag{3}$$

In the meta-network training stage, to simulate channel selection decisions, an embedding dimension $\phi = E$ is randomly sampled at each training step and adopted to the MHSA layer. Similarly, the channel-variable MLP consists of two linear projections and one non-linear activation function, where the first linear projection is used to expand the channels by a factor of γ [2], [38], and the second linear projection is used to restore the channels, which can be represented as:

$$\texttt{MLP}(\mathbf{x};\gamma) = \texttt{GeLU}\left(\mathbf{x}\left(\mathbf{W}^{\texttt{UP}}_\gamma\right)^\mathsf{T}\right)\left(\mathbf{W}^{\texttt{DOWN}}_\gamma\right)^\mathsf{T}. \tag{4}$$

Similar to Eq. 3, we slice the first γ times input channels from the full weights $\mathbf{W}^{\texttt{UP}}$ and $\mathbf{W}^{\texttt{DOWN}}$ to obtain $\mathbf{W}^{\texttt{UP}}_\gamma$ and $\mathbf{W}^{\texttt{DOWN}}_\gamma$. The trained meta-network provides the basis our method to adjust the embedding dimension. In the following parts, we will describe how to optimize token number and token length through PPO selector and achieve the best trade-off between sample-wise accuracy and computational complexity.

C. Sample-wise Joint Optimization with Lightweight PPO Agent

After obtaining the meta-network, we consider building the sample-wise selector capable of jointly optimizing token number N and model channels C. This selector, integral to the framework, should meet two paramount criteria: *(1) lightweight* and *(2) sample-effective*. The former necessitates minimal consumption of storage overheads and computational resources to circumvent parity with direct classification. Conversely, the latter indicates effective learning of decision-making processes to maximize the utilization of valid information in both the model parameters and the data, while minimizing resource consumption. In this paper, we model the token and channel reduction process as a Markov decision process and employ PPO accordingly.

Formulation of Joint Token and Architecture Optimization. To reduce the decision cost, we apply the selector within each Transformer group, consisting of every three Transformer blocks, to determine the token optimization (*i.e.*, pruning, merging, or pruning-merging) ratio and the network structures

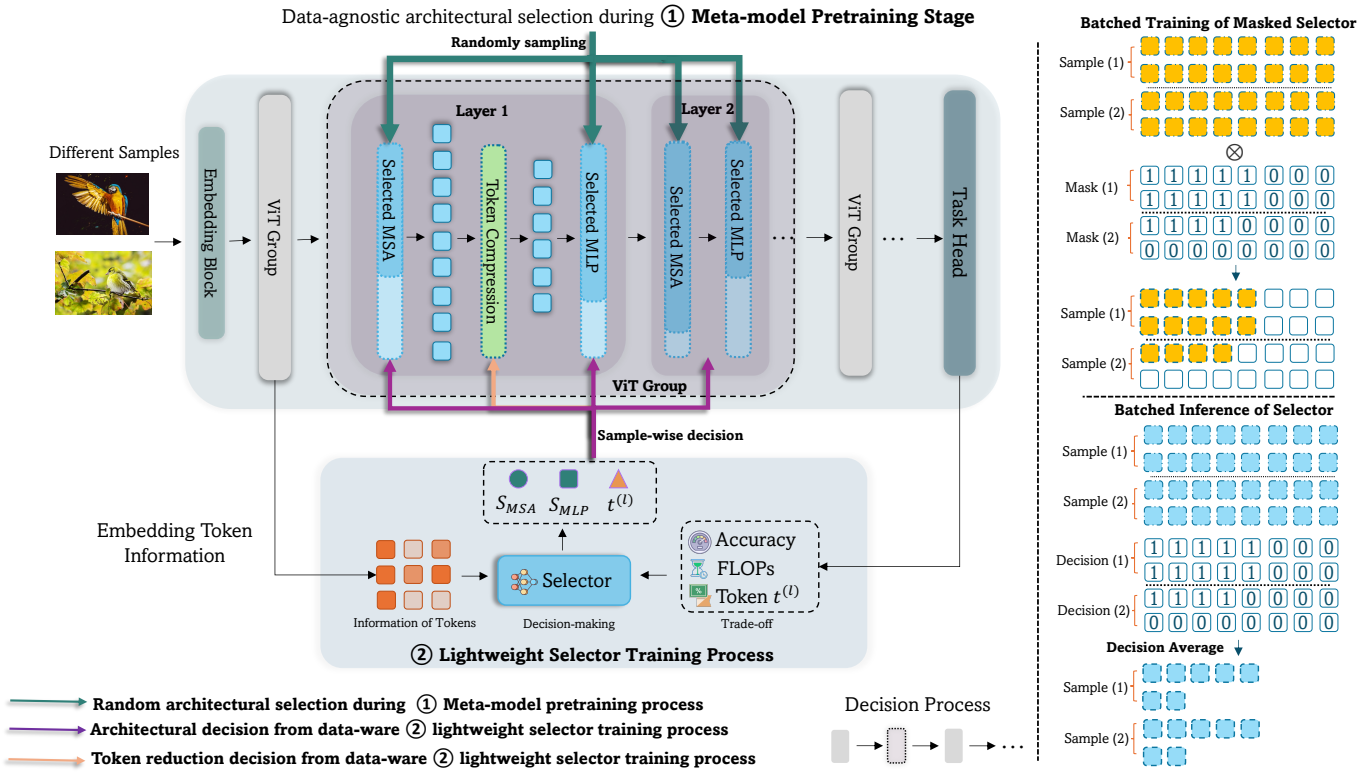


Fig. 3. **The framework of PRANCE.** ***Left:*** The training of PRANCE consists of two stages: (1) Meta-model Pretraining. The meta-network is trained using the weight-sharing mechanism, where the smaller channels are subsets of the large channels, to support the variable channels. To simulate the variable channel decisions, a configuration is randomly selected for the MHSA layer and MLP layer in each training step. In this stage, we do not perform token optimization. (2) Sample-wise architecture-data joint optimization. After convergence of the meta-network, we freeze the meta-network and train the PPO selector using the "Result-to-Go" mechanism. In this stage, the PPO selector will jointly make the decisions for channel reduction of the MHSA layer and MLP layer, along with the decision of token reduction. ***Right:*** We adopt a sample-wise masking mechanism for supporting batched training of the selector, where the decisions are generated in the form of 0-1 mask and applied on the corresponding inputs (e.g., tokens, channels) using Hadamard product accordingly, to ensure dimensional consistency. During inference, the sample-wise mask can be replaced by averaging the decisions of each batch.

for the subsequent blocks, tailored to each sample. Therefore, the selector is formulated as:

$$\mathbf{s}^{(l)},\ \mathbf{t}^{(l)} = \mathcal{F}_{\texttt{Selector}}\left(\mathbf{O}^{(l-1)}\right), \tag{5}$$

where l is the group index, $\mathbf{t}^{(l)}$ represents the token keep ratio for l-th Transformer group, $\mathbf{s}^{(l)}$ denotes the structures decision of l-th Transformer group, and the $\mathbf{O}^{(l-1)}$ is the feature extracted by the $(i-1)$-th Transformer group, representing the abstracted data information up to the current block in the ViTs. According to previous works [19], [20], [23], <CLS> token, $\mathbf{Q}$ (query), $\mathbf{K}$ (key), $\mathbf{V}$ (value), and the output of Self-Attention $\mathbf{X}$ can be used. For simplicity, we omit the notations of dimensions for these matrices.

Token optimization consists of two steps: *(1) token importance ranking* and *(2) token optimization*. In the first step, tokens are sorted by their contributions to the task, so that a specific token optimization method can be applied in the second step according to the token keep ratio. For token importance ranking, as <CLS> progressively aggregates the task-specific (*e.g.*, classification) global information, the inner product of class token <CLS> and other tokens reflect the importance of different tokens. Hence, we leverage this mechanism for token ranking to measure whether a token is important to the input samples. To get accurate informative information, we directly use the first MHSA layer $\texttt{MHSA}_1^{(l)}$ in the l-th Transformer group to extract the importance vector $\mathbf{a}_{cls}^{(l)}$ for the output of last Transformer group $\mathbf{Y}^{(l-1)}$ to avoid an additional matrix multiplication:

$$\mathbf{X}^{(l)} = \texttt{Sort}\left(\texttt{MHSA}_1^{(l)}\left(\mathbf{Y}^{(l-1)}; \mathbf{s}^{(l)}\right), \alpha_{cls}^{(l)}\right) \quad \text{where}$$
$$\alpha_{cls}^{(l)} = \texttt{Softmax}\left(\frac{\mathbf{q}_{cls}^{(l)} \cdot \left(\mathbf{K}^{(l)}\right)^{\mathsf{T}}}{\sqrt{C^{(l)}}}\right)\mathbf{V}^{(l)}, \tag{6}$$

where $\mathbf{q}_{cls}^{(l)}$ is the query of the class tokens. Therefore, $\alpha_{cls}^{(l)}$ of Eq. (6) is actually a vector of the output $\texttt{MHSA}_1^{(l)}(\mathbf{Y}^{(l-1)})$. $\texttt{Sort}(\cdot)$ is the sorting function that can arrange tokens in descending order based on $\mathbf{A}_{cls}^{l}$.

After preprocessing the tokens, we consider three representative token reduction strategies to obtain the tokens for the remaining MHSA layers and MLP layers in l-th group: *(i) pruning*, *(ii) merging*, and *(iii) pruning-then-merging*. For token pruning, unimportant tokens will be discarded for each sample according to $\mathbf{t}^{(l)}$ [20]:

$$\mathbf{X}^{(l)} = \mathbf{X}^{(l)}\left[:\texttt{round}\left(N^{(l)} \times \mathbf{t}^{(l)}\right), :\right]. \tag{7}$$

For token merging, the sorted tokens will be divided into two categories based on the token keep ratio $\mathbf{t}^{(l)}$: important tokens

$\mathbf{X}_{\mathbf{im}}^{(l)}$ and unimportant tokens $\mathbf{X}_{\mathbf{un}}^{(l)}$:

$$\mathbf{X}_{\mathbf{im}}^{(l)} = \mathbf{X}^{(l)}\left[:, \texttt{round}\left(N^{(l)} \times \mathbf{t}^{(l)}\right), :\right], \quad \text{and} \tag{8}$$

$$\mathbf{X}_{\mathbf{un}}^{(l)} = \mathbf{X}^{(l)}\left[\texttt{round}\left(N^{(l)} \times \mathbf{t}^{(l)}\right):, :\right]. \tag{9}$$

Subsequently, each unimportant token $\mathbf{X}_{\mathbf{un}}^{i}$ will be merged into an optimal important token $\mathbf{X}_{\mathbf{im}}^{j}$ that is most similar to it, to formulate a new $\mathbf{X}^{(l)}$ for next layers:

$$\mathbf{X}^{(l)} = \{\mathbf{X}^{m}\}_{m=0}^{M}, \quad \text{where} \quad \mathbf{X}_{\mathbf{im}}^{m} = \mathbf{X}_{\mathbf{im}}^{m} + \mathbf{X}_{\mathbf{un}}^{n}, \tag{10}$$

where $M = \texttt{round}\left(N^{(l)} \times \mathbf{t}^{(l)}\right)$ represents the number of kept tokens, m, n are the indexes which achieve maximal cosine similarity $\cos(\boldsymbol{\theta}_{mn})$, which is calculated by:

$$\cos(\boldsymbol{\theta}_{mn}) = \frac{\mathbf{X}_{\mathbf{un}}^{m} \cdot \mathbf{X}_{\mathbf{im}}^{n}}{||\mathbf{X}_{\mathbf{un}}^{m}|| \cdot ||\mathbf{X}_{\mathbf{im}}^{n}||}. \tag{11}$$

For pruning and merging, we adopt the pruning-then-merging [25] scheme. Specifically, the token keep ratio is divided into a token pruning ratio $\mathbf{t}_{prune}^{(l)}$ along with a token merging ratio $\mathbf{t}_{merge}^{(l)}$, i.e., $\mathbf{t}^{(l)} = \left\{\mathbf{t}_{prune}^{(l)}, \mathbf{t}_{merge}^{(l)}\right\}$.

After the token optimization, the remaining tokens will go through the latter Transformer blocks within this group, with the architectural decisions based on $\mathbf{s}^{(l)}$.

Lightweight Selector Modeling. For the proposed PRANCE, it is crucial to construct a lightweight yet high-performing selector capable of sample-aware optimization for both model structure and tokens, ensuring accuracy and FLOPs within a large optimization space. However, it is non-trivial to learn the solution via conventional supervised learning, as it is difficult to collect substantial labeled data. Considering the outstanding performance of RL in various decision-making tasks such as gaming [39], [40], control [41], [42], combinatorial optimization [43], and data augmentation [44], we decided to use the PPO algorithm to effectively optimize the selector in such a large decision space. Among serious RL methods, PPO [45] is one of the on-policy algorithms with relatively stable performance and wide applicability, which has even been applied to many popular language models to improve their effectiveness [46].

In this paper, we adopt a continuous output strategy instead of the hybrid output strategy. This approach treats both network structure optimization and token optimization as continuous action optimization problems. Through preliminary experiments, we found that the continuous output strategy provides more stable training and achieves better performance than the hybrid output strategy, even with a lighter network structure.

The training of PPO consists of two components: an actor network for generating the decisions for architectural optimization and token reduction, and a value network $\mathcal{V}$ for predicting the value of the current state. During the training process, the value network will evaluate value of the current state $\mathbf{O}^{(l)}$ first, then the actor network will generate the corresponding policy $\{\mathbf{s}^{(l)}, \mathbf{t}^{(l)}\}$ based on the evaluation to reach the states with higher value. During inference, only the actor network is used to serve as the selector $\mathcal{F}_{\texttt{Selector}}$.

The training process involves limiting the magnitude of each policy update to prevent drastic changes, by adopting a clipped loss function to control the difference between the new and old policies and optimizing this loss function to improve policy performance. Specifically, for the l-th Transformer group, we use GAE (Generalized Advantage Estimation) [47] to calculate the advantage function, and the objective of the value network $\mathcal{V}$ is to predict the state value with utmost accuracy, thus we employ a loss function derived from Temporal-difference methods:

$$\begin{cases} \mathbf{L}_{V}^{(l)} = \left(\mathbf{r}^{(l)} + \gamma\mathcal{V}(\mathbf{O}^{(l+1)}) - \mathcal{V}(\mathbf{O}^{(l)})\right)^2/B \\ \hat{\mathbf{A}}_{GAE(\gamma,\lambda)}^{(l)} = \sum_{i=0}^{\infty}(\gamma\lambda)^i \mathbf{L}_{\mathcal{V}}^{(l+i)} \end{cases}, \tag{12}$$

where B is batch size, γ is the discount factor, $\mathbf{r}^{(l)}$ is the reward of l-th Transformer group, and $\mathbf{O}^{(l)}$ is the state information (the input of selector in Eq. (5)) of the l-th Transformer group, $\hat{\mathbf{A}}^{(l)}$ denotes the estimation of the advantage function. Then the optimal policies can be obtained by maximizing the rewards of the sequence:

$$\begin{aligned} \mathbf{L}^{CLIP}(\theta) = &-\hat{\mathbb{E}}^{(l)}\Big[\min\left(r^{(l)}(\theta)\hat{\mathbf{A}}^{(l)}, \Phi\right) \\ &-\delta\mathbb{E}_{a\sim\pi}\left[-\log\left(\pi(\mathbf{a}^{(l)}|\mathbf{O}^{(l)})\right)\right]\Big], \\ \text{where} \quad \Phi = &\texttt{Clip}\left(r^{(l)}(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{\mathbf{A}}^{(l)}, \end{aligned} \tag{13}$$

π is the policy of the actor network A, δ is the policy entropy ratio, which will facilitate PPO's exploration of the action space, $r^{(l)}(\theta)$ is the import sampling ratio of current policy and old policy:

$$r^{(l)}(\theta) = \frac{\pi_{\theta}(\mathbf{a}^{(l)}|\mathbf{O}^{(l)})}{\pi_{\theta_{\text{old}}}(\mathbf{a}^{(l)}|\mathbf{O}^{(l)})}. \tag{14}$$

Besides, to achieve better results, we optimized PPO with techniques such as Advantage Normalization [48], Gradient Clip, and Orthogonal Initialization.

Sequence model of "Result-to-Go". In general, it is crucial for RL algorithm to capture the impact of each decision on the final result during training [22], [49]. However, the ViTs model encounters a *delayed-return* scenario where the final result becomes available only after passing through all the ViTs blocks during the sequence decision process, while it requires multiple selector decisions throughout this process. Inspired by [50], [51], we conduct a timely return model named "Result-to-Go", which is shown in Fig 4. First of all, the maximum structural parameters and 100% token keep ratio are assigned to the model. During the forward propagation process, the PPO selector will optimize the activated channels and useful tokens in the current ViTs group, and the model will continue to the end to acquire the classification result without changing the parameters of other groups. Following this paradigm, the activated channels and the useful token numbers of each ViTs group will be modified gradually and progressively. It is worth mentioning that the changes of model structures and token numbers are tailored to specific samples, rather than being applied at a coarse-grained task level.

In this way, although obtaining the result of a single layer-by-layer decision requires multiple rounds of forward propagation, we can get timely effects for each decision. During the training process, the parameters of ViTs are held constant, and

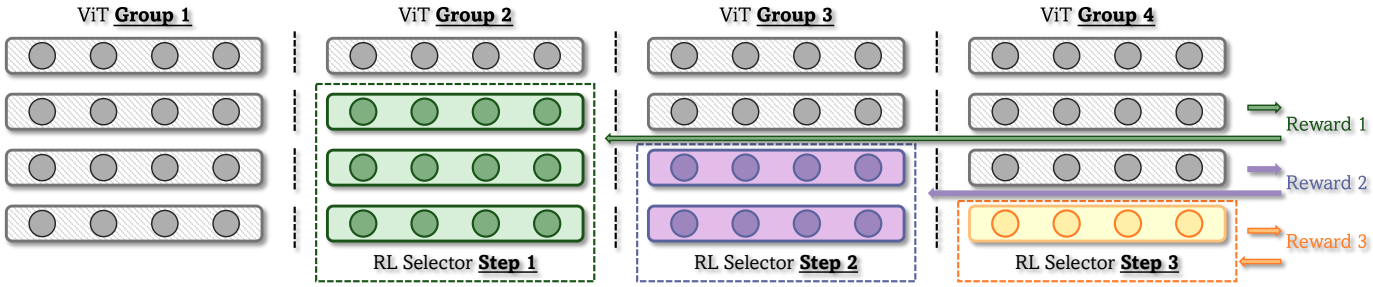


Fig. 4. **The workflow of "Result-to-Go".** This mechanism is only used for training the selector. To receive immediate feedback for each decision, the meta-network is divided into multiple groups. Initially, the meta-network is set to the maximum channel number for all groups. The selector then optimizes the model channels and tokens numbers for a single group at a time, allowing the meta-network to run to the end and obtain immediate feedback. Since the meta-network is fixed, its inference process can be viewed as a Markov decision process, allowing the selector to modify the structure of the meta-network groups one by one.

the paradigm of "Result-to-Go" is utilized to train the PPO. However, it will be disabled during inference to ensure model efficiency.

Reward function. During PPO training, the reward function varies for each sample to achieve a sample-wise selector. Specifically, it consists of three parts: *(1) Top-1 accuracy reward* $\boldsymbol{r}_{acc}$, *(2) FLOPs penalty* $\boldsymbol{r}_f$ and *(3) token optimization penalty* $\boldsymbol{r}_t$. Although we have obtained the meta-network with high accuracy on the dataset, the classification results of individual samples still vary between correct and incorrect. Providing feedback to the selector based on it may introduce some disturbance: even if PPO generates optimal structural parameters and token optimization rates, incorrect feedback may still occur due to limitations of model's classification level. In order to overcome it, we maintain two sets of results during training: the classification result $\mathbf{y}_t$ of the supernet and the dynamically optimized classification result $\mathbf{y}$ involving the selector. If $\mathbf{y}$ is the same as $\mathbf{y}_t$, then $\mathbf{y}$ is set to 1. In this way, the selector can obtain smooth and accurate feedback. The total reward function is:

$$\mathbf{r} = \mathbf{r}_{acc} - a_f \mathbf{f} - a_t \mathbf{t}_\mathbf{r} \quad \text{where} \quad \mathbf{r}_{\texttt{acc}} = \begin{cases} \mathbf{1}, & \texttt{if } \mathbf{y} = \mathbf{y}_t \\ \mathbf{r}_{\texttt{acc}}, & \texttt{otherwise} \end{cases}, \tag{15}$$

where a_f and a_t represent the punishment factor of FLOPs ratio and token number ratio, respectively. They are utilized to control the trade-off between accuracy and FLOPs, and encourage token optimization. $\mathbf{f}$ is the FLOPs of the model, $\mathbf{t}_\mathbf{r}$ represents the token keep ratio, $\mathbf{r}_f = -a_f \mathbf{f}$ and $\mathbf{r}_t = -a_t \mathbf{t}_\mathbf{r}$. It is worth noting that $\mathbf{r}_t$ is important. Due to the minor impact of modifying the network structure on accuracy, while token reduction has a greater impact on the result, PPO may tend to retain tokens, thereby falling into a local optimum. This phenomenon has also been observed in our experiments.

Actor function. The actor function is constructed from the perspective of RL environment modeling. In terms of architectural optimization, the selector has a limited set of selections, namely the channels of MHSA and MLP layer, to choose from, making it a discrete decision problem. In terms of data optimization, the selector has to generate a token optimization ratio $\mathbf{t} \in [0, 1]$, which leads to a continuous decision problem. Such a mixed-decision problem is hard to optimize because it significantly improves the complexity of PPO action space, we therefore transform the architectural optimization problem into a continuous decision problem.

For l-th Transformer group, with each Transformer group has K MHSA layers and K MLP layers, the architectural decision involves $\mathbf{s}^{(l)} = \{s^{(l)}_{\texttt{MLP}}, s^{(l)}_{\texttt{MHSA}}\}$. Specifically, $\mathbf{s}^{(l)}_{\texttt{MLP}} = \{s^{(l)}_{\texttt{MLP},k}\}^{K}_{k=0}$ denotes the decided MLP ratios for the K MLP layers in this Transformer group, where $\forall k \in K, s^{(l)}_{\texttt{MLP},k} \in [0, 1]$, and $\mathbf{s}^{(l)}_{\texttt{MHSA}} = \{s^{(l)}_{\texttt{MHSA},k}\}^{K}_{k=0}$ denotes the decided embedding dimension ratios for the K MHSA layers in this Transformer group, $\forall k \in K, s^{(l)}_{\texttt{MHSA},k} \in [0, 1]$. Suppose $E_{\texttt{MHSA}}$ and $E_{\texttt{MLP}}$ represent the number of architectural candidates of the MHSA and MLP layer, the index of the architectural parameters of this group can be calculated by:

$$\mathbf{i}^{(l)}_{\texttt{MHSA}} = \left\{ \texttt{round}\left(s^{(l)}_{\texttt{MHSA},k} * E_{\texttt{MHSA}} \right) \right\}^{K}_{k=0}, \tag{16}$$

$$\mathbf{i}^{(l)}_{\texttt{MLP}} = \left\{ \texttt{round}\left(s^{(l)}_{\texttt{MLP},k} * E_{\texttt{MLP}} \right) \right\}^{K}_{k=0}. \tag{17}$$

Accordingly, we can obtain the architectural parameters based on these indexes.

The token $\mathbf{t}^{(l)}$ represents the token pruning keep ratio, token merging keep ratio, or a combination of both, denoted as $\mathbf{t}^{(l)} = \left\{ \mathbf{t}^{(l)}_{prune}, \mathbf{t}^{(l)}_{merge} \right\}$. Depending on the selected token optimization policies, it serves as a basis for conducting sample-specific token optimization.

Masked Selector Training. It is infeasible to train the selector on GPUs, due to diverse token numbers and embedding dimensions across different samples make the ViTs cannot perform batched parallel computations. Therefore, we leverage masking to enable parallel computation within existing frameworks for training the selector.

Specifically, suppose B, N, and C represent the batch size, maximum token number, and maximum embedding dimension, respectively, therefore $i \in [0, B]$, $j \in [0, N]$, and $k \in [0, C]$ denote the indices of batch size, number of tokens, and number of channels. The MLP Mask $\mathbf{M}^{L}_{ijk}$ is applied to the MLP layers by padding zero to the channels beyond the selected embedding dimension for different samples to align the dimension, and Token Mask $\mathbf{M}^{\mathsf{T}}_{ijk}$ add masks to tokens of

different samples:

$$\mathbf{M}^L_{ijk} = \begin{cases} 1, & \texttt{if } k < \mathbf{d_e}[i] \\ 0, & \texttt{otherwise} \end{cases}, \quad \mathbf{X} = \mathbf{X} \odot \mathbf{M}^L_{ijk}, \tag{18}$$

$$\mathbf{M}^{\mathsf{T}}_{ijk} = \begin{cases} 1, & \texttt{if } j < \mathbf{d_t}[i] \\ 0, & \texttt{otherwise} \end{cases}, \quad \mathbf{X} = \mathbf{X} \odot \mathbf{M}^{\mathsf{T}}_{ijk}, \tag{19}$$

where $\mathbf{d_e}$ and $\mathbf{d_t}$ represent vectors containing selected embedding dimensions and token numbers of samples, respectively, and i denotes the sample number. $\odot$ represents the element-wise multiplication.

When it comes to Token Merging, the matrices $\mathbf{X_{im}}$ and $\mathbf{X_{un}}$, containing important and unimportant tokens respectively, can be constructed through Token Mask, where the masked tokens are set to ∞ instead of 0. Then the cosine similarity matrix is $\mathbf{S} = \mathbf{X_{un}} \odot \mathbf{X^{\mathsf{T}}_{im}}$. After setting ∞ to $-\infty$, the useful information will be concentrated in the lower-left corner of matrix $\mathbf{S}$, enabling sample-wise token merging based on it. This approach also relaxes the restriction of merging up to 50% of the tokens by ToMe [23].

For LayerNorm which requires the computations of mean and standard deviation, we suppose the mask token matrix is $\mathbf{X}$ and the mask matrix is $\mathbf{M}^L_{ijk}$. Initially, we perform mean filling for the parts of the token matrix that are masked, to mitigate the potential impact of mean calculation on the performance of the LayerNorm function. The valid sum value and the number of them can be calculated as follows:

$$\mathbf{X}_{\text{mask}}[i,j] = \sum_{k=1}^{C} \mathbf{X}_{ijk}, \quad \mathbf{S}_X[i,j] = \sum_{k=1}^{C} \mathbf{M}^L_{ijk}. \tag{20}$$

The mean value will be filled into the token matrix:

$$\mathbf{X}_{\text{fill}} = \mathbf{X} + (1 - \mathbf{M}^L) \cdot \mathbf{X}_{\text{mean}} \quad \text{where} \quad \mathbf{X}_{\text{mean}} = \frac{\mathbf{X}_{\text{mask}}[i,j]}{\mathbf{S}_X[i,j]}. \tag{21}$$

Next, the token matrix filled with mean value can be used to do sample-wise Layernorm:

$$\mathbf{X} = \texttt{Layernorm}(\mathbf{X}_{\text{fill}}) \cdot \frac{\sqrt{N}}{\sqrt{\mathbf{S}_X}}. \tag{22}$$

IV. Experimental Result

In this section, we conduct extensive experiments to demonstrate the effectiveness of PRANCE. All the experiments are conducted on ImageNet, and we report results for three different-sized models: ViT-Tiny, ViT-Small, and ViT-Base. Specifically, these models have roughly 1.2 GFLOPS, 5G FLOPs, and 20 GFLOPs, respectively. All experiments besides the selector training are performed on NVIDIA A100 GPUs with 40G memory with PyTorch training system, whereas we train the selector using a single NVIDIA RTX3090 GPU.

A. Experimental Settings.

Meta-network. The meta-networks are conducted based on the architecture of DeiT [52], which consists of a total of 12 Transformer blocks. The embedding dimension and MLP dimension of the Attention layer are configurable. Besides, in order to further provide more selections for the PPO selector, we increase the structural complexity of the MLP by selecting multiples of the embedding dimension. The detailed model information can be found in Tab. I. Specifically, we train the models for 500 epochs using the AdamW optimizer and adopt mixed-precision training, and the first 20 epochs are used for warm-up with the learning rate set to 1×10^{-6}. We use the cosine learning rate scheduler for training. The initial learning rate is set to 5×10^{-4}, the decay rate is 0.1, and the minimum learning rate is 1×10^{-5}. The training process will use about 1.5 GPU days.

TABLE I
Setting of the Meta-network.

Model	Embedding Dim	MLP Ratio	Heads	Depth
Tiny	{176, 192, 216, 240}	{2, 4, 6}	3	12
Small	{320, 352, 368, 384, 400, 416}	{2, 3.5, 5}	7	12
Base	{672, 696, 720, 744, 768}	{2, 3.5, 5}	12	12

The embedding dimension and MLP dimension of the Attention block are set as optional structural dimensions, and the structural space is expanded by setting the MLP dimension option to be the multiple of the embedding dimension.

PPO Selector. To obtain accurate information for support decisions, we enable the selector after the first Transformer group. In other words, the token and architecture will not be changed during the inference of the first Transformer group. For each Transformer group $l \in \{2, ..., L\}$ that is to be decided, the outputs of the selector include the token keep ratio $\mathbf{t}^{(l)}$, and the architectural decision $\mathbf{s}^{(l)} = \{\mathbf{s}^{(l)}_{\texttt{MLP}}, \mathbf{s}^{(l)}_{\texttt{MSA}}\}$, where $\mathbf{s}^{(l)}_{\texttt{MLP}}$ represents the MLP expansion ratios and $\mathbf{s}^{(l)}_{\texttt{MSA}}$ represents the embedding dimension of MSA layer. The $\mathbf{t}^{(l)} \in [0, 1]$ will be directly used to optimize the tokens of the current group. The actor network and critic network of the PPO selector each consist of 3 fully connected layers, with a state dimension of 197 and a hidden dimension of 256. The actor dimension depends on token optimization strategies: it is set to 7 for pruning and merging, 8 for pruning-then-merging. The learning rate of actor network and critic network are 1×10^{-4} and 5×10^{-3}, respectively. The token punish ratio a_t is 0.2. The FLOPs punish ratio a_f is determined by the peak FLOPs of the meta-network, with a punishment range of [0.2, 0.6]. For different model scales, the specific ranges are set as follows: [0.02, 0.04] for tiny models, [0.008, 0.015] for small models, and [0.002, 0.006] for base models. The interval is 0.005 for both the tiny and base models, and 0.002 for the small model. Besides, the selector is trained for 1 epoch using 50000 images sampled from the training dataset within 30 minutes. During inference, decisions can be averaged directly over the batch dimension without performance degradation.

Finetuning. We follow the fine-tuning settings of DynamicViT [19]. Specifically, we finetune the meta-network with 30 epochs using the cosine learning rate scheduler and do not perform warm-up, we use an external CNN teacher to further improve the performance. During fine-tuning, we freeze the PPO selector. The initial learning rate is set to 2×10^{-5}, the minimize learning rate is 2×10^{-6}, the weight decay is 1×10^{-6}. The mixup is disabled to improve the convergence.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3605239

TABLE II
THE MAIN RESULTS OF PRANCE.

Model	Method	Top-1 Acc. (%)	FLOPs (G)	Token Keep Rate	Architectural Optimization	Token Optimization Strategy
Tiny	DeiT-Tiny	72.20%	1.20	100%	-	N/A
	SAViT [53]	70.72% (↓1.48)	0.90 (↓25%)	N/A	✓	N/A
	UPDP [54]	70.12% (↓2.08)	0.90 (↓25%)	N/A	✓	N/A
	A-ViT [33]	71.00% (↓1.20)	0.80 (↓33%)	-	×	Token Pruning
	DynamicViT [19]	70.90% (↓1.30)	0.90 (↓25%)	70.00% (Fixed)	×	Token Pruning
	S^2ViTE [55]	70.12% (↓2.08)	0.90 (↓25%)	30.00% (Fixed)	×	Token Pruning
	SPViT [56]	72.10% (↓0.10)	0.90 (↓25%)	34.30% (Fixed)	×	Token Pruning
	Evo-ViT [34]	72.00% (↓0.20)	0.73 (↓39%)	25.00% (Fixed)	×	Token Pruning
	ToMe [23]	71.27% (↓0.93)	0.90 (↓25%)	70.00% (Fixed)	×	Token Merging
	Ours	**72.38%** (↑**0.18**)	0.87 (↓28%)	25.00% (Learned)	✓	Token Pruning
	Ours	**72.81%** (↑**0.61**)	0.96 (↓20%)	53.00% (Learned)	✓	Token Merging
	Ours	**73.31%** (↑**1.11**)	0.87 (↓28%)	33.00% (Learned)	✓	Token P + M
Small	DeiT-Small	79.90%	4.70	100%	-	N/A
	A-ViT [33]	78.60% (↓1.30)	3.60 (↓23%)	-	×	Token Pruning
	DynamicViT [19]	79.32% (↓0.58)	2.90 (↓38%)	34.30% (Fixed)	×	Token Pruning
	Evo-ViT [34]	79.40% (↓0.50)	3.00 (↓36%)	25.00% (Fixed)	×	Token Pruning
	SPViT [56]	79.80% (↓0.10)	3.86 (↓18%)	34.30% (Fixed)	×	Token Pruning
	EViT [20]	79.50% (↓0.40)	3.00 (↓36%)	34.30% (Fixed)	×	Token Pruning
	S^2ViTE [55]	79.22% (↓0.68)	3.20 (↓32%)	40.00% (Fixed)	×	Token Pruning
	GTP-ViT [57]	78.80% (↓1.10)	2.90 (↓38%)	14.29% (Fixed)	×	Token Merging
	ToMe [23]	78.85% (↓1.05)	2.67 (↓43%)	27.00% (Fixed)	×	Token Merging
	DiffRate [25]	79.47% (↓0.43)	2.85 (↓39%)	48.73% (Learned)	×	Token P + M
	LTMP [58]	79.30% (↓0.60)	2.70 (↓43%)	-	×	Token P + M
	Ours	**80.02%** (↑**0.12**)	2.75 (↓41%)	36.00% (Learned)	✓	Token Pruning
	Ours	**80.17%** (↑**0.27**)	2.85 (↓39%)	38.00% (Learned)	✓	Token Merging
	Ours	**80.19%** (↑**0.29**)	2.96 (↓37%)	33.00% (Learned)	✓	Token P + M
	ViT-Slimming [59]	77.90% (↓2.00)	2.30 (↓51%)	N/A	✓	N/A
	X-Pruner [59]	78.93% (↓0.97)	2.40 (↓49%)	N/A	✓	N/A
	EViT [20]	78.50% (↓1.40)	2.30 (↓51%)	12.50% (Fixed)	×	Token Pruning
	DynamicViT [19]	77.50% (↓2.40)	2.20 (↓53%)	50.00% (Fixed)	×	Token Pruning
	GTP-ViT [57]	78.30% (↓1.60)	2.60 (↓45%)	14.29% (Fixed)	×	Token Merging
	Diffrate [25]	78.81% (↓1.09)	2.40 (↓49%)	33.50% (Learned)	×	Token P + M
	Ours	**79.40%** (↓0.50)	2.28 (↓53%)	13.00% (Learned)	✓	Token Pruning
	Ours	**79.98%** (↑**0.08**)	2.38 (↓49%)	18.00% (Learned)	✓	Token Merging
	Ours	**79.98%** (↑**0.08**)	2.55 (↓46%)	33.00% (Learned)	✓	Token P + M
Base	DeiT-Base	81.80%	18.00	100%	-	N/A
	AS-DeiT-B [60]	81.40% (↓0.40)	11.20 (↓38%)	34.30% (Fixed)	×	Token Pruning
	DynamicViT [19]	81.43% (↓0.37)	11.40 (↓37%)	34.30% (Fixed)	×	Token Pruning
	EViT [20]	80.00% (↓1.80)	8.70 (↓52%)	12.50% (Fixed)	×	Token Pruning
	EViT [20]	81.30% (↓0.50)	11.50 (↓36%)	34.30% (Fixed)	×	Token Pruning
	Evo-ViT [34]	81.30% (↓0.50)	11.31 (↓37%)	25.00% (Fixed)	×	Token Pruning
	GTP-ViT [57]	80.70% (↓1.10)	11.00 (↓38%)	32.65% (Fixed)	×	Token Merging
	ToMe [23]	80.29% (↓1.51)	10.57 (↓41%)	27.00% (Fixed)	×	Token Merging
	Diffrate [25]	79.73% (↓2.07)	9.42 (↓48%)	30.46% (Learned)	×	Token P + M
	Diffrate [25]	81.27% (↓0.53)	11.61 (↓36%)	47.72% (Learned)	×	Token P + M
	LTMP [58]	78.80% (↓3.00)	9.00 (↓50%)	-	×	Token P + M
	Ours	**81.49%** (↓0.31)	10.90 (↓39%)	51.00% (Learned)	✓	Token Pruning
	Ours	**81.77%** (↓0.03)	10.59 (↓41%)	65.00% (Learned)	✓	Token Merging
	Ours	**81.52%** (↓0.28)	9.54 (↓47%)	67.00% (Learned)	✓	Token P + M

We present the performance comparison of PRANCE with various SOTA methods across three model sizes: Tiny, Small, and Base, with the results of three different token optimization methods: Pruning, Merging, and Pruning-then-Merging (Token P+M).

This process will be completed within 0.5 GPU days.

B. Main results

Tab. II shows the results of the PRANCE across three different-sized models. The Top-1 accuracy, FLOPs, and token keep rate are reported, along with comparative analyses against other methods. Note that the token keep rate refers to the percentage of tokens retained after three rounds of optimization. One can see that PRANCE significantly reduces the FLOPs while maintaining exceptionally high accuracy. Under the same FLOPs constraints, PRANCE outperforms existing lightweight SOTA models. Notably, for ViT-Tiny and ViT-Small, our models even slightly surpass the accuracy of original models, showcasing that reducing the redundancies of ViTs can improve their generalization. Specifically, for ViT-Tiny, our model achieves approximately 1% higher Top-1 accuracy than the base model while reducing FLOPs by about 30%. For ViT-Small, with FLOPs reductions of approximately 40% and 50%, the Top-1 accuracy surpasses the base model by about 0.3% and 0.1%, respectively. For ViT-Base, with a

reduction of approximately 40% in FLOPs, the Top-1 accuracy is only 0.03% lower than that of the base model.

Overall, PRANCE is an efficient sample-wise inference method that optimizes both model structural dimensions and data dimensions simultaneously, enabling optimal results with minimal FLOPs. Moreover, by adaptively optimizing each sample through data optimization methods such as token pruning, token merging, token pruning-then-merging, and structural optimization methods such as channel selection, we can significantly reduce the model size while maintaining or even improving the model's accuracy. On the one hand, the optimization of both the model structure and data dimensions by PRANCE contributes significantly. On the other hand, the powerful adaptive decision-making capability of the PPO selector ensures that even the simplest optimization methods can be effectively applied to each sample, thus avoiding to use more complex lightweight techniques such as token clustering. Tab. III presents the results of PRANCE employing different token optimization strategies across various model settings. The results of each token optimization strategy at different optimization levels are reported. First of all, for models of various scales, PRANCE employing pruning, merging, and pruning-then-merging optimization strategies can maintain high accuracy with significantly reduced FLOPs. These results significantly surpass various lightweight SOTA models and even meet or exceed the performance of DeiT, demonstrating the powerful lightweight capability of our model and its excellent performance across different compression levels. Moreover, the results show that the three token optimization methods—pruning, merging, and pruning-then-merging—can all achieve excellent performance. This is different from the conclusions of more complex token optimization methods. With the help of the PPO selector, our model achieves a more reasonable token optimization ratio for different samples, even though it only uses sample token optimization methods.

Meanwhile, token merging outperforms pruning at extremely low FLOPs. For instance, for ViT-Small with 1.96G FLOPs, the accuracy can still reach 79.29%, for ViT-Base with 7.09G FLOPs, the accuracy can still reach 81.29%. From the perspective of joint optimization of data and model structure, the token keep rate is the result of data dimension optimization, while the model channels represent the structural dimension optimization. For models of the same scale, the effect of data optimization and model optimization are coupled: increasing data or enhancing channel dimensions both improve accuracy, and there is a complementary relationship between them. For example, PRANCE at the Base scale can achieve an accuracy of 81.51% by retaining only 8% of the tokens, or it can also achieve a similar accuracy of 81.49% by retaining 56% of the tokens. What's more, although PRANCE can surpass DeiT in accuracy, the margin by which PRANCE exceeds DeiT decreases as the scale of DeiT increases. And PRANCE struggles to surpass DeiT in accuracy at the base scale. This indicates that for ImageNet, the model's scale transitions from being insufficient to becoming overly large. At the Base scale, the training data is insufficient for the model to learn more.

TABLE III
THE DETAILS OF PRANCE

Model	Top-1 Acc. (%)	FLOPs(G)	Token Keep Rate	Type
DeiT-Tiny	72.20%	1.20	100%	
Tiny	72.38% (↑**0.18**)	0.87 (↓28%)	25%	Pruning
	73.07% (↑**0.87**)	0.97 (↓19%)	25%	Pruning
	73.55% (↑**1.35**)	1.07 (↓11%)	60%	Pruning
	71.39% (↓0.81)	0.87 (↓28%)	51%	Merging
	72.81% (↑**0.61**)	0.96 (↓20%)	53%	Merging
	73.05% (↑**0.85**)	1.03 (↓14%)	53%	Merging
	72.36% (↑**0.16**)	0.71 (↓41%)	8%	P + M
	73.31% (↑**1.11**)	0.87 (↓28%)	33%	P + M
	73.77% (↑**1.57**)	0.94 (↓22%)	41%	P + M
	74.41% (↑**2.21**)	1.03 (↓14%)	33%	P + M
DeiT-Small	79.90%	4.70	100%	
Small	78.93% (↓0.97)	2.07 (↓56%)	11%	Pruning
	79.40% (↓0.50)	2.28 (↓51%)	13%	Pruning
	79.59% (↓0.31)	2.59 (↓45%)	30%	Pruning
	80.12% (↑**0.22**)	2.84 (↓40%)	36%	Pruning
	80.25% (↑**0.35**)	3.17 (↓33%)	36%	Pruning
	79.29% (↓0.61)	1.96 (↓58%)	18%	Merging
	79.98% (↑**0.08**)	2.38 (↓49%)	18%	Merging
	80.01% (↑**0.11**)	2.40 (↓49%)	18%	Merging
	80.05% (↑**0.15**)	2.58 (↓45%)	38%	Merging
	80.17% (↑**0.27**)	2.85 (↓39%)	38%	Merging
	80.21% (↑**0.31**)	3.05 (↓35%)	38%	Merging
	79.42% (↓0.48)	2.10 (↓55%)	29%	P + M
	79.71% (↓0.19)	2.30 (↓51%)	29%	P + M
	79.98% (↑**0.08**)	2.55 (↓46%)	33%	P + M
	80.06% (↑**0.16**)	2.61 (↓44%)	33%	P + M
	80.15% (↑**0.25**)	3.25 (↓31%)	33%	P + M
DeiT-Base	81.80%	18.00	100%	
Base	81.29% (↓0.51)	9.60 (↓47%)	51%	Pruning
	81.43% (↓0.37)	10.41 (↓42%)	51%	Pruning
	81.49% (↓0.31)	11.34 (↓37%)	56%	Pruning
	81.29% (↓0.51)	7.09 (↓61%)	36%	Merging
	81.42% (↓0.38)	7.64 (↓58%)	36%	Merging
	81.51% (↓0.29)	9.30 (↓48%)	7%	Merging
	81.77% (↓0.03)	10.59 (↓41%)	65%	Merging
	80.24% (↓1.56)	7.43 (↓59%)	27%	P + M
	81.23% (↓0.57)	8.20 (↓54%)	47%	P + M
	81.52% (↓0.28)	9.54 (↓47%)	67%	P + M
	81.62% (↓0.18)	11.66 (↓35%)	73%	P + M

The results of PRANCE in jointly optimizing the model structure with three token optimization strategies: Pruning, Merging, and pruning-then-merging, across three different-sized models, including ViT-Tiny, ViT-Small, and ViT-Base. Besides, for each optimization scheme, results under multiple FLOPs constraints are reported to demonstrate the effectiveness of PRANCE.

C. Analysis

One-step selection VS multi-step selection. For the sample-wise joint optimization problem of data and model structure, the most intuitive approach is to generate all necessary parameters for lightweight inference at once, based on the initial samples. However, this approach treats ViTs as a black box and constructs a substantially large decision space with up to around 10^{14} combinations. Another way is to follow the paradigm of RL by modeling the problem as a Markov process. PRANCE divides ViTs into multiple groups and allowing the PPO selector to make decisions in stages. In this way, the action space is reduced to just 7 or 8 dimensions, which not only significantly reduces the difficulty of joint optimization but also helps the selector grasp the token features at different stages of the inference process.

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3605239

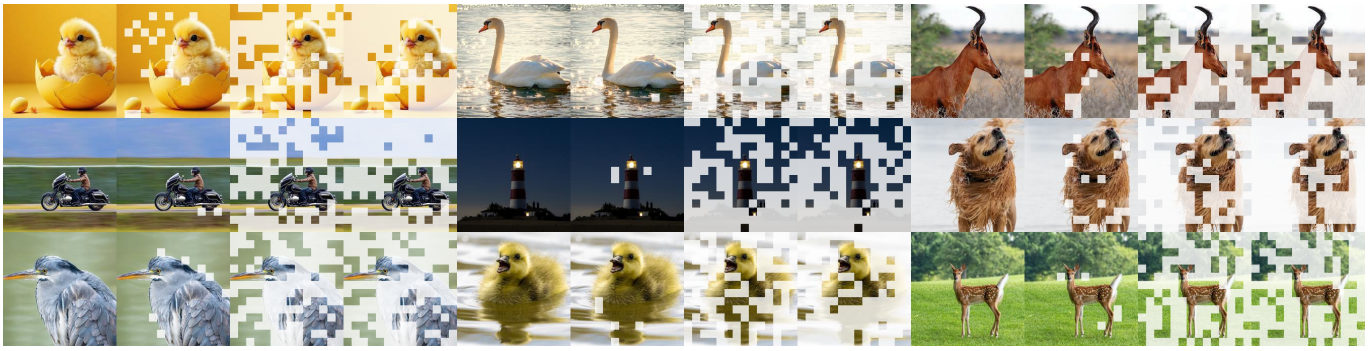


Fig. 5. **Visualization of token pruning in different transformer groups.** PRANCE effectively identifies and retains important tokens while removing unimportant ones to reduce the complexity of ViTs. Besides, our framrwork prefers to retain tokens in the early stages and optimize a large number of low-information tokens in the later stages.

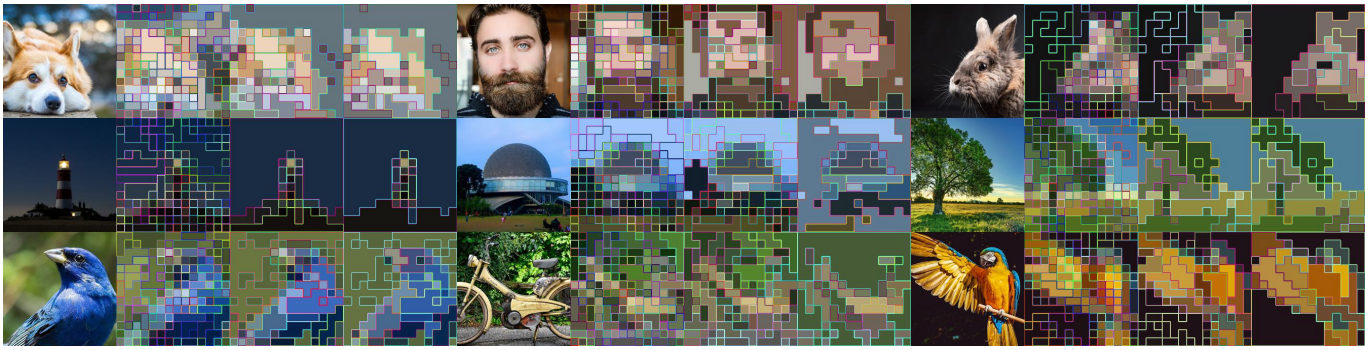


Fig. 6. **Visualization of token merging in different transformer groups.** PRANCE can merge similar tokens based on their importance, retaining tokens with higher information.

TABLE IV
THE RESULT OF ONE-STEP DECISION AND MULTI-STEP DECISION.

Decision Type	Model	Top-1 Acc. (%)	FLOPs(G)	Token Keep Rate
Once	Tiny	62.44%	1.02	30% ([0.5, 1])
	Small	64.00%	3.18	36% ([0.5, 1])
	Base	75.05%	10.9	42% ([0.5, 1])
Step by step	Tiny	73.07%	0.97	25% ([0, 1])
	Small	80.12%	2.84	36% ([0, 1])
	Base	81.43%	10.41	51% ([0, 1])

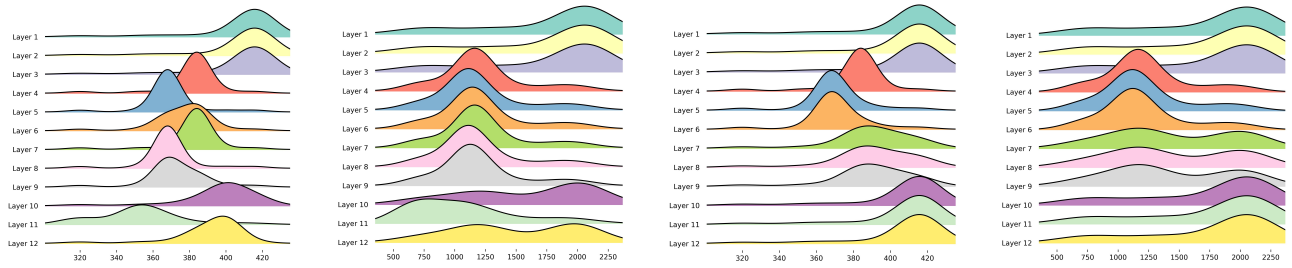
The intervals marked in blue indicate the range of token keep ratios for single-step decisions. Compared with one-step solution, the multi-step apporach can better capture the trade-off between accuracy with model structure and tokens.

Tab. IV shows the results of the above two model ways. The one-step selector fails to effectively learn the importance of tokens and model structure, resulting in poor performance. Conversely, the multi-step selector, modeled based on PRANCE, can better capture the trade-off between accuracy with model structure and tokens, leading to better accuracy. Specifically, to avoid performance collapse, the token keep ratio for single-step decisions is set to [0.5, 1], while the PRANCE ranges from [0, 1].

Impact of penalization on token optimization strategies. An interesting phenomenon was observed during training the PRANCE: without penalizing the token optimization terms, the selector tends to retain all tokens, and obtain a high-precision lightweight model by optimizing the model structure. The results of the comparative experiments are shown in Tab. V. As the model scale increases, the selector tends to keep more tokens. In the Tiny scale, a good balance can be achieved even without adding the penalty factor a_t. However, in the Small and Base scales, the proportion of retained tokens is around 80% or higher, sometimes even reaching 100%. The reason may be that adjusting the structure of the meta-network has a smaller impact on accuracy compared to optimizing tokens. Besides, this phenomenon becomes more pronounced as the model scale increases, which demonstrates that the model has learned effective information from the data. As the scale increases, the less information is required from the data. Unfortunately, merely optimizing the model structure while retaining a large number of tokens results in a suboptimal state. Although the Top-1 accuracy of the model is maintained, the computational complexity remains high. By adding penalty terms to the reward function, PRANCE effectively achieves a balance between the number of tokens and the model structure, reducing FLOPs to a lower level while maintaining Top-1 accuracy.

The effect of different features on Selector. Inspired by existing lightweighting efforts, we believe that the intermediate features of the trained ViTs can effectively reflect data characteristics. Therefore, we attempted to optimize the model structure and tokens on a sample-wise basis, separately utilizing the Class Token $\mathbf{X}$, $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ and $\mathbf{QKV}$ feature matrices in each ViTs' block. The experiments are conducted on three models of different scales, and the results are presented in Tab.

Fig. 7. **Visualization of token pruning-then-merging (P+M) in different transformer groups.** The light, translucent parts represent the pruned tokens, while the colored blocks represent the merged tokens. PRANCE can prune the least informative background tokens based on the complexity of the image, then merge tokens with less information, and retain tokens with higher information content.



(a) Dimensional Distribution of Embedding Channels of the Model for the "Bird" Category.

(b) Dimensional Distribution of MLP of the Model for the "Bird" Category.

(c) Dimensional Distribution of Embedding Channels of the Model for the "Church" Category.

(d) Dimensional Distribution of MLP of the Model for the "Church" Category.

Fig. 8. **Dimensional distributions of different categories on ImageNet.** The category "Church" contains complex architectural information with high information, while the category "Bird" is a photograph with a simple background and less information. One can see that PRANCE can dynamically select model channels based on the complexity of the image, achieving the highest possible Top-1 accuracy with the lowest possible FLOPs.

TABLE V
THE EFFECT OF PENALIZATION ON TOKEN OPTIMIZATION.

Model	Reward State	Type	Top-1 Acc. (%)	FLOPs(G)	Token Keep Rate
Tiny	Penalty	Pruning	66.39%	0.99	38%
		Merging	68.85%	1.18	32%
		P + M	71.59%	1.18	60%
	Raw	Pruning	71.92%	1.18	60%
		Merging	70.51%	0.92	53%
		P + M	71.15%	0.89	46%
Small	Penalty	Pruning	74.10%	2.72	13%
		Merging	76.97%	1.98	9%
		P + M	75.28%	2.36	29%
	Raw	Pruning	77.57%	3.78	96%
		Merging	78.44%	4.17	97%
		P + M	76.92%	3.92	73%
Base	Penalty	Pruning	78.33%	9.18	59%
		Merging	74.61%	8.31	51%
		P + M	79.99%	10.7	30%
	Raw	Pruning	81.40%	15.18	100%
		Merging	79.90%	13.79	88%
		P + M	80.90%	14.18	93%

The impact of token penalties on various ViTs models and token optimization strategies. Due to the significant impact of optimizing tokens on accuracy, the selector tends to optimize the model structure and retain tokens. This situation is significantly mitigated after adding token penalties.

TABLE VI
THE EFFECT OF DIFFERENT FEATURES ON SELECTOR.

Model	Input	Top-1 Acc. (%)	FLOPs(G)	Token Keep Rate
Tiny	**Q**	72.01%	0.91	42%
	K	72.38%	0.87	25%
	V	70.09%	0.95	53%
	QKV	72.62%	0.93	23%
Small	**Q**	79.63%	2.72	30%
	K	80.12%	2.84	36%
	V	77.97%	3.11	32%
	QKV	80.27%	3.02	29%
Base	**Q**	81.32%	10.59	64%
	K	81.29%	9.60	51%
	V	80.90%	11.36	89%
	QKV	81.24%	10.34	31%

The results of different features as inputs for the selector on ViT-Tiny, Small, and Base models. The $<$ CLS $>$ token is discarded due to its high dimensionality and excessive computational overhead. The K matrix performs relatively the best across models of different scales.

VI. Due to the maximum embedding dimensions of 240, 416, and 768 for the three model scales, which result in excessive training costs and suboptimal accuracy, the Class Token $\mathbf{X}$ was not adopted. The $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, $\mathbf{QKV}$ matrices perform well on different model scales, with the $\mathbf{K}$ matrix showing the most significant effect. QKV and K achieve similar results, but the selector's computation cost is 3 times higher when using the QKV matrix. Therefore, we believe that using the K matrix is the best choice.

The impact of different features on token merging. To improve computational efficiency, we draw on token merging methods from ToMe [23] and Diffrate [25], exploring the

TABLE VII
THE IMPACT OF DIFFERENT FEATURES ON TOKEN MERGING.

Model	Input	Type	Top-1 Acc. (%)	FLOPs(G)	Token Keep Rate
Tiny	K	Merging	71.90%	0.92	62%
		P + M	73.58%	0.99	40%
	X	Merging	72.81%	0.96	53%
		P + M	74.36%	1.00	33%
Small	K	Merging	78.72%	2.64	28%
		P + M	78.82%	2.90	44%
	X	Merging	80.17%	2.85	38%
		P + M	80.19%	2.96	33%
Base	K	Merging	79.70%	9.12	54%
		P + M	79.37%	8.22	56%
	X	Merging	81.57%	9.64	65%
		P + M	77.02%	11.85	74%

The impact of various selector inputs on token merging and pruning-then-merging across ViT models (before fine-tuning). The performance of adopting $\mathbf{X}$ is significantly better than that of matrix $\mathbf{K}$, and this difference becomes more pronounced as the model size increases.

TABLE VIII
THE EFFECT OF SMOOTH ACCURACY.

Model	Reward State	Type	Top-1 Acc. (%)	FLOPs(G)	Token Keep Rate
Tiny	Raw	Pruning	67.67%	1.01	59%
		Merging	71.91%	0.96	26%
		P + M	73.40%	1.18	31%
	Smooth	Pruning	70.62%	1.07	60%
		Merging	71.07%	0.92	35%
		P + M	72.67%	1.01	42%
Small	Raw	Pruning	72.99%	3.26	76%
		Merging	75.38%	2.15	35%
		P + M	68.74%	2.97	56%
	Smooth	Pruning	74.10%	2.70	13%
		Merging	75.48%	2.07	24%
		P + M	75.28%	2.36	29%
Base	Raw	Pruning	81.04%	14.50	55%
		Merging	79.46%	12.65	75%
		P + M	79.62%	12.52	94%
	Smooth	Pruning	81.08%	11.50	51%
		Merging	79.22%	10.45	71%
		P + M	79.99%	10.69	30%

The impact of the smoothing function on model performance across various ViTs models and various token optimization strategies before fine-tuning. The smoothing function has little impact on the Tiny model but significantly affects the Small and Base models. On the other hand, its impact is more pronounced on the pruning strategy compared to the merging strategy.

effects of feature matrices $\mathbf{K}$ and $\mathbf{X}$ in token importance ranking and merging. The results are shown in Tab. VII. To intuitively demonstrate the impact of the two evaluation metrics on token optimization, the results before fine-tuning are presented. Overall, matrix $\mathbf{X}$ has a more favorable impact on the results compared to matrix $\mathbf{K}$, and this effect becomes increasingly pronounced as the model size grows. Besides, for both token merging and token pruning-then-merging, the impact of the two matrices is similar. A possible reason is that, compared to matrix $\mathbf{K}$, matrix $\mathbf{X}$ not only includes information about token optimization, but also provides more intuitive information about model channel optimization. From the training perspective, obtaining a high-performance PPO selector based on the $\mathbf{X}$ matrix is much easier than using the $\mathbf{K}$ matrix, which often requires a very challenging parameter tuning process.

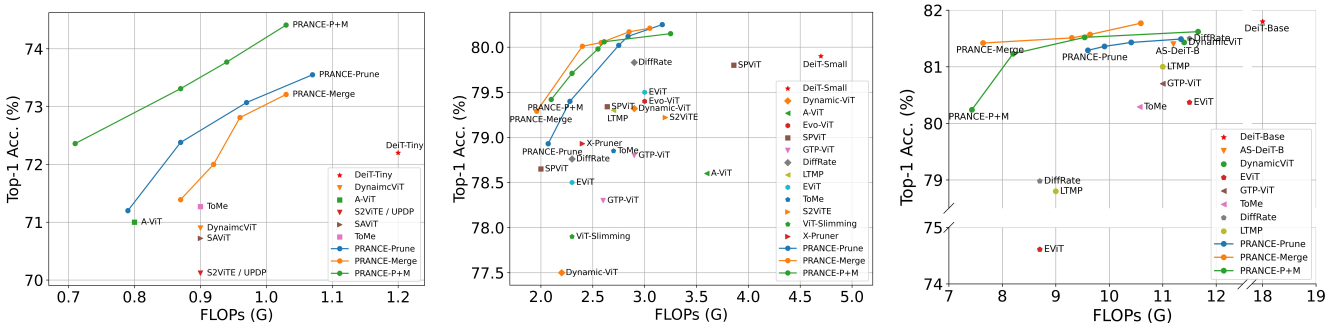
The effect of smooth accuracy. The Top-1 accuracy is modified by Eq. (15) to avoid the influence of meta-network performance. Tab. VIII shows the comparison results. Note that these results are obtained before fine-tuning, which can more clearly demonstrate the impact of the smoothing function. Overall, the performance of using the smoothing function is superior to not using it, and this effect becomes increasingly pronounced as the model scale increases. At the Tiny scale, smooth function has no significant impact. While at the Small scale, the impact is only evident in the pruning strategy, causing both the pruning and pruning-then-merging strategies to maintain high Top-1 accuracy, but at the cost of increased FLOPs. Ultimately, at the Base scale, this impact extends to all token optimization strategies.

The inference efficiency. To ensure inference efficiency, we evaluate the inference latency under full load conditions (maximum batch size) and the throughput of PRANCE. Two scenarios is considered: (1) Resource-limited edge-device with a NVIDIA GTX 970 GPU (only 4GB memory), and (2) routine device with a NVIDIA RTX 2080Ti GPU (11GB memory). The results are shown in Tab. IX and Tab. X. PRANCE show remarkable efficiency and becomes increasingly pronounced with larger model sizes and larger batch size. On resource-limited device GTX 970, PRANCE demonstrates significant efficiency improvement across models of three different scales, reducing latency by over 25%, with a peak reduction of 38%, all without any loss in accuracy. On RTX 2080ti GPU, PRANCE achieves about 1.3∼1.7 times throughout improvements across model and token optimization strategies. Besides, the PPO selector almost has no effect on the inference time. In a single inference process, the selector will be called 3 times, taking a total of about 1∼2 ms. The added delay is significantly smaller than the time saved from inference acceleration.

What's more, we can also find that different hardware requires different token optimization strategies. For example, on the NVIDIA GTX 970 GPU, Token Pruning-then-Merging strategy consistently achieves the best latency reduction across all models (up to 38%). Meanwhile, on the NVIDIA RTX 2080ti GPU, Token Pruning strategy excels in throughput for smaller models (1.72×), while Token Merging is more effective for larger models like the Base model (1.57×). This highlights that the advantage of PRANCE that supporting flexible token optimization strategies for various hardware-model combinations, with pruning favoring lightweight models on high-performance GPUs and merging or combined approaches better for larger models.

Visualization. In this part, some visualization results will be shown. First of all, Fig. 5, Fig. 6, Fig. 7 present the step-by-step visualization results of PRANCE on different samples using the pruning, merging, and pruning-then-merging strategies, respectively. All three strategies can dynamically optimize different numbers of tokens based on the complexity of the sample at various stages of model inference, retaining important tokens and removing unimportant ones. While there are still some differences between them. The pruning strategy

This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3605239



(a) ViT-Tiny (b) ViT-Small (c) ViT-Base

Fig. 9. **Accuracy-FLOPs performance.** Across different model scales, PRANCE achieves higher Top-1 accuracy with lower FLOPs, surpassing various SOTA methods and even exceeding DeiT at the Tiny and Small scales. On the other hand, as the model scale increases, the margin by which PRANCE surpasses DeiT becomes smaller. This indicates that the model is learning more, and the amount of data becomes insufficient at the Base scale.

TABLE IX
THE INFERENCE LATENCY OF THE PRANCE.

Model	Uncompressed (ms)	Type	Compressed (ms)	Selector (ms)
Tiny	121.25 ± 5.26	Pruning	86.63 ± 5.26 (↓29%)	1.58 ± 0.23
		Merging	87.62 ± 5.72 (↓28%)	1.67 ± 0.58
		P + M	84.90 ± 4.53 (↓30%)	1.68 ± 0.30
Small	131.89 ± 7.44	Pruning	85.63 ± 4.34 (↓35%)	1.57 ± 0.22
		Merging	84.94 ± 5.31 (↓36%)	1.78 ± 0.43
		P + M	82.37 ± 4.78 (↓38%)	1.55 ± 0.19
Base	111.15 ± 4.24	Pruning	81.64 ± 4.35 (↓27%)	1.64 ± 0.24
		Merging	82.16 ± 4.23 (↓26%)	1.65 ± 0.25
		P + M	68.44 ± 4.91 (↓38%)	1.70 ± 0.28

As the model size and batch size increases, the acceleration effect of PRANCE becomes increasingly pronounced. The latency of all models are reduced by over 25%, with a peak reduction of up to 38%. Besides, the selector adds almost no overhead to inference time.

TABLE X
THROUGHPUT MEASUREMENT (IMG/SEC) OF PRANCE.

Model	Uncompressed (img/sec)	Type	Compressed (img/sec)
Tiny	1002.58	Pruning	1469.57 (×1.46)
		Merging	1333.67 (×1.33)
		P + M	1403.97 (×1.40)
Small	485.16	Pruning	836.18 (×1.72)
		Merging	688.47 (×1.41)
		P + M	638.55 (×1.31)
Base	197.36	Pruning	259.22 (×1.31)
		Merging	310.38 (×1.57)
		P + M	286.60 (×1.45)

PRANCE achieves substantial improvements in model throughput, delivering over 1.3× speedup across models of different scales, with a peak improvement of 1.7×. This highlights its ability to support flexible token optimization strategies across diverse hardware-model combinations.

primarily optimizes tokens in the later stages, specifically at the 6th and 9th blocks, while retaining more tokens at the 3rd block. The pruning-then-merging strategy, on the other hand, tends to optimize tokens in the early stages, with the lightest optimization at the 9th layer. In contrast, the merging strategy has a more balanced optimization process. This phenomenon is closely related to the optimization of the model structure. When the model has acquired sufficient information, it tends to optimize more tokens. Conversely, the model prefer to use as many tokens as possible to capture sample information.

Fig. 8 shows the distribution of channels in the embedding layers and MLP layers. To ensure a fair comparison, we fixed the first three layers to their maximum structure, while the remaining nine layers were optimized. Samples in the "Bird" category are relatively simple: the main subject of the image is complex, but the background is very simple. In contrast, samples in the "Church" category are more complex, with the entire image filled with intricate architectural details. It can be observed that for complex samples, PRANCE tends to use more channels, while for simple samples, the number of activated channels is significantly reduced. It demonstrates that PRANCE can dynamically adjust the model's complexity based on different samples. What's more, the dynamic adjustment of the model structure by PRANCE is more evident in samples of varying complexity rather than in samples of different categories. This is because samples from different categories can contain both simple and complex images.

Fig. 9 illustrates the correlation curves between FLOPs and Top-1 accuracy under three different scales. The models closer to the top-left corner in the figure exhibit better performance. Overall, PRANCE achieves excellent results across different model scales by employing various token optimization methods. It not only surpasses the vast majority of SOTA lightweight algorithms but also matches or exceeds the performance of the baseline model DeiT with significantly lower FLOPs. On the other hand, the different token optimization methods can all achieve good results, with no significant differences in their effectiveness. This is related to the selector's collaborative optimization of the model structure and tokens, as well as the complementary information in different token optimization modes.

V. CONCLUSION

In this paper, we propose the PRANCE framework for ViTs compression that optimizes both the architecture (model channels) and data (number of tokens). To this end, we pre-train a meta-network that supports variable channels and then model the inference process of ViTs as a Markov decision process, using PPO as the selector, and propose a matching "Result-to-Go" training mechanism. PRANCE supports the joint opti-

mization of the model structure and three different token optimization methods: pruning, merging, and pruning-merging, all of which yield good results. Extensive experiments have demonstrated the outstanding performance of this framework, surpassing existing SOTA methods and showcasing significant potential for widespread applications.

ACKNOWLEDGMENTS

This work was supported by National Key Research and Development Project of China (Grant No. 2023YFF0905502), National Natural Science Foundation of China(Grant No. 92467204 and 62472249), Shenzhen Science and Technology Program (Grant No. JCYJ20220818101014030 and KJZD20240903102300001) and National Natural Science Foundation of China (Grant No. 62402264).

REFERENCES

[1] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, "A survey on vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 87–110, 2022.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[3] M. Gehrig and D. Scaramuzza, "Recurrent vision transformers for object detection with event cameras," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 2023, pp. 13 884–13 893.

[4] Z. Dai, B. Cai, Y. Lin, and J. Chen, "Unsupervised pre-training for detection transformers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 12 772–12 782, 2023.

[5] M. K. H. Thisanke, L. A. C. Deshan, K. Chamith, S. Seneviratne, R. Vidanaarachchi, and D. Herath, "Semantic segmentation using vision transformers: A survey," *Eng. Appl. Artif. Intell.*, vol. 126, p. 106669, 2023.

[6] H. Ding, C. Liu, S. Wang, and X. Jiang, "Vlt: Vision-language transformer and query generation for referring segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 7900–7916, 2023.

[7] Y. Wang, X. Chen, L. Cao, W. Huang, F. Sun, and Y. Wang, "Multimodal token fusion for vision transformers," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 2022, pp. 12 176–12 185.

[8] J. Deng, Z. Yang, D. Liu, T. Chen, W. Zhou, Y. Zhang, H. Li, and W. Ouyang, "Transvg++: End-to-end visual grounding with language conditioned vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13 636–13 652, 2023.

[9] M. Zhu, Y. Tang, and K. Han, "Vision transformer pruning," *arXiv preprint arXiv:2104.08500*, 2021.

[10] F. Yu, K. Huang, M. Wang, Y. Cheng, W. Chu, and L. Cui, "Width & depth pruning for vision transformers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 3143–3151.

[11] Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, and W. Gao, "Post-training quantization for vision transformer," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 092–28 103, 2021.

[12] C. Tang, K. Ouyang, Z. Wang, Y. Zhu, W. Ji, Y. Wang, and W. Zhu, "Mixed-precision neural network quantization via learned layer-wise importance," in *European Conference on Computer Vision*. Springer, 2022, pp. 259–275.

[13] C. Tang, Y. Meng, J. Jiang, S. Xie, R. Lu, X. Ma, Z. Wang, and W. Zhu, "Retraining-free model quantization via one-shot weight-coupling learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 15 855–15 865.

[14] M. Chen, H. Peng, J. Fu, and H. Ling, "Autoformer: Searching transformers for visual recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 270–12 280.

[15] C. Tang, L. L. Zhang, H. Jiang, J. Xu, T. Cao, Q. Zhang, Y. Yang, Z. Wang, and M. Yang, "Elasticvit: Conflict-aware supernet training for deploying fast vision transformer on diverse mobile devices," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 5829–5840.

[16] J. Yu, P. Jin, H. Liu, G. Bender, P.-J. Kindermans, M. Tan, T. Huang, X. Song, R. Pang, and Q. Le, "Bignas: Scaling up neural architecture search with big single-stage models," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*. Springer, 2020, pp. 702–717.

[17] H. Yang, H. Yin, P. Molchanov, H. Li, and J. Kautz, "Nvit: Vision transformer compression and parameter redistribution," *CoRR*, vol. abs/2110.04869, 2021.

[18] D. Wang, M. Li, C. Gong, and V. Chandra, "Attentivenas: Improving neural architecture search via attentive sampling," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6418–6427.

[19] Y. Rao, Z. Liu, W. Zhao, J. Zhou, and J. Lu, "Dynamic spatial sparsification for efficient vision transformers and convolutional neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 883–10 897, 2023.

[20] Y. Liang, C. Ge, Z. Tong, Y. Song, J. Wang, and P. Xie, "Not all patches are what you need: Expediting vision transformers via token reorganizations," in *International Conference on Learning Representations*, 2022.

[21] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[22] H. Wang, J. Fan, Z. Chen, H. Li, W. Liu, T. Liu, Q. Dai, Y. Wang, Z. Dong, and R. Tang, "Optimal transport for treatment effect estimation," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[23] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your vit but faster," *arXiv preprint arXiv:2210.09461*, 2022.

[24] D. Bolya and J. Hoffman, "Token merging for fast stable diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4598–4602.

[25] M. Chen, W. Shao, P. Xu, M. Lin, K. Zhang, F. Chao, R. Ji, Y. Qiao, and P. Luo, "Diffrate: Differentiable compression rate for efficient vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17 164–17 174.

[26] S. Long, Z. Zhao, J. Pi, S. Wang, and J. Wang, "Beyond attentive tokens: Incorporating token importance and diversity for efficient vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 10 334–10 343.

[27] H. Wang, B. Dedhia, and N. K. Jha, "Zero-tprune: Zero-shot token pruning through leveraging of the attention graph in pre-trained transformers," *arXiv preprint arXiv:2305.17328*, 2023.

[28] A. Chavan, Z. Shen, Z. Liu, Z. Liu, K.-T. Cheng, and E. P. Xing, "Vision transformer slimming: Multi-dimension searching in continuous optimization space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4931–4941.

[29] Y. Lin, T. Zhang, P. Sun, Z. Li, and S. Zhou, "Fq-vit: Post-training quantization for fully quantized vision transformer," *arXiv preprint arXiv:2111.13824*, 2021.

[30] C. Tang, H. Zhai, K. Ouyang, Z. Wang, Y. Zhu, and W. Zhu, "Arbitrary bit-width network: A joint layer-wise quantization and adaptive inference approach," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 2899–2908.

[31] X. Liu, H. Peng, N. Zheng, Y. Yang, H. Hu, and Y. Yuan, "Efficientvit: Memory efficient vision transformer with cascaded group attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 420–14 430.

[32] Y. Liang, C. Ge, Z. Tong, Y. Song, J. Wang, and P. Xie, "Not all patches are what you need: Expediting vision transformers via token reorganizations," *arXiv preprint arXiv:2202.07800*, 2022.

[33] H. Yin, A. Vahdat, J. M. Alvarez, A. Mallya, J. Kautz, and P. Molchanov, "A-vit: Adaptive tokens for efficient vision transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 809–10 818.

[34] Y. Xu, Z. Zhang, M. Zhang, K. Sheng, K. Li, W. Dong, L. Zhang, C. Xu, and X. Sun, "Evo-vit: Slow-fast token evolution for dynamic vision transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 2964–2972.

[35] J. B. Haurum, M. Madadi, S. Escalera, and T. B. Moeslund, "Multi-scale hybrid vision transformer and sinkhorn tokenizer for sewer defect classification," *Automation in Construction*, vol. 144, p. 104614, 2022.

[36] D. Marin, J.-H. R. Chang, A. Ranjan, A. Prabhu, M. Rastegari, and O. Tuzel, "Token pooling in vision transformers for image classification," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 12–21.

[37] C. Renggli, A. S. Pinto, N. Houlsby, B. Mustafa, J. Puigcerver, and C. Riquelme, "Learning to merge tokens in vision transformers," *arXiv preprint arXiv:2202.12015*, 2022.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[39] J. Fan, Y. Zhuang, Y. Liu, J. Hao, B. Wang, J. Zhu, H. Wang, and S. Xia, "Learnable behavior control: Breaking atari human world records via sample-efficient behavior selection," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[40] J. Fan, "A review for deep reinforcement learning in atari: Benchmarks, challenges, and solutions," *CoRR*, vol. abs/2112.04145, 2021.

[41] Y. Li, Z. Liu, G. Lan, M. Sader, and Z. Chen, "A ddpg-based solution for optimal consensus of continuous-time linear multi-agent systems," *Science China Technological Sciences*, vol. 66, no. 8, pp. 2441–2453, 2023.

[42] Z. Liu, Y. Li, G. Lan, and Z. Chen, "A novel data-driven model-free synchronization protocol for discrete-time multi-agent systems via td3 based algorithm," *Knowledge-Based Systems*, vol. 287, p. 111430, 2024.

[43] C. Wang, Z. Yu, S. McAleer, T. Yu, and Y. Yang, "Asp: Learn a universal neural solver!" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 6, pp. 4102–4114, 2024.

[44] J. Fan and C. Xiao, "Generalized data distribution iteration," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 2022, pp. 6103–6184.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[46] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022.

[47] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[48] G. Tucker, S. Bhupatiraju, S. Gu, R. Turner, Z. Ghahramani, and S. Levine, "The mirage of action-dependent baselines in reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 5015–5024.

[49] J. Fan, C. Xiao, and Y. Huang, "GDI: rethinking what makes reinforcement learning different from supervised learning," *CoRR*, vol. abs/2106.06232, 2021.

[50] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[51] R. Lin, Y. Li, X. Feng, Z. Zhang, X. H. W. Fung, H. Zhang, J. Wang, Y. Du, and Y. Yang, "Contextual transformer for offline meta reinforcement learning," *arXiv preprint arXiv:2211.08016*, 2022.

[52] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[53] C. Zheng, K. Zhang, Z. Yang, W. Tan, J. Xiao, Y. Ren, S. Pu *et al.*, "Savit: Structure-aware vision transformer pruning via collaborative optimization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9010–9023, 2022.

[54] J. Liu, D. Tang, Y. Huang, L. Zhang, X. Zeng, D. Li, M. Lu, J. Peng, Y. Wang, F. Jiang *et al.*, "Updp: A unified progressive depth pruner for cnn and vision transformer," *arXiv preprint arXiv:2401.06426*, 2024.

[55] T. Chen, Y. Cheng, Z. Gan, L. Yuan, L. Zhang, and Z. Wang, "Chasing sparsity in vision transformers: An end-to-end exploration," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 974–19 988, 2021.

[56] Z. Kong, P. Dong, X. Ma, X. Meng, W. Niu, M. Sun, X. Shen, G. Yuan, B. Ren, H. Tang *et al.*, "Spvit: Enabling faster vision transformers via latency-aware soft token pruning," in *European conference on computer vision*. Springer, 2022, pp. 620–640.

[57] X. Xu, S. Wang, Y. Chen, Y. Zheng, Z. Wei, and J. Liu, "Gtp-vit: Efficient vision transformers via graph-based token propagation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 86–95.

[58] M. Bonnaerens and J. Dambre, "Learned thresholds token merging and pruning for vision transformers," *Transactions on Machine Learning Research*, 2023.

[59] A. Chavan, Z. Shen, Z. Liu, Z. Liu, K.-T. Cheng, and E. P. Xing, "Vision transformer slimming: Multi-dimension searching in continuous optimization space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4931–4941.

[60] X. Liu, T. Wu, and G. Guo, "Adaptive sparse vit: Towards learnable adaptive token pruning by fully exploiting self-attention," *arXiv preprint arXiv:2209.13802*, 2022.

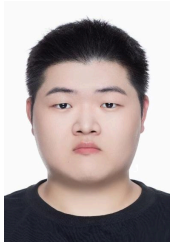
This article has been accepted for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2025.3605239

VI. Biography Section

Ye Li is currently pursuing the Ph.D. degree in the Department of Computer Science and Technology at Tsinghua University. He received the M.S. degree from the College of Artificial Intelligence, Nankai University in 2023. His research interests include efficient deep learning, edge computing, and deep reinforcement learning.

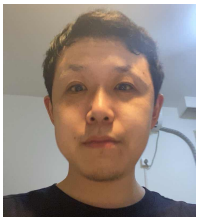
Chen Tang is currently a Ph.D. student at the Multimedia Laboratory (MMLab), The Chinese University of Hong Kong. He received his Master's degree in Computer Technology from Tsinghua University. Before CUHK, he worked as a research assistant in the Department of Computer Science and Technology at Tsinghua University. He was a research intern in the System and Networking Research Group of Microsoft Research Asia (MSRA). He has published several papers on top-tier conferences, including CVPR, ICCV, ECCV, ACM MM, and EMNLP. His research interests are in efficient deep learning (e.g., quantization, sparsity), hardware-aware neural architecture search, and dynamic neural networks.

Yuan Meng (Member, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China. She is currently an assistant professor with the media and network lab, THDCST, Tsinghua. Her research interests include multimedia edge intelligence and model compression, especially OOD generalization, as reflected in her publications on top-tier journals and conferences, including CPVR, ICCV, ACM MM, and INFOCOM.



Jiajun Fan received his Bachelor's degree from Nankai University in Tianjin, China, and his Master's degree from Tsinghua University in Beijing, China. He is currently pursuing a Ph.D. at the Siebel Center for Computer Science within The Grainger College of Engineering at the University of Illinois Urbana-Champaign, Champaign, IL, USA. Jiajun has published several papers in top-tier conferences, including ICLR, ICML, and NeurIPS, and has served as a reviewer for KDD, ICML, AAAI, ICLR, and NeurIPS. His research interests include deep reinforcement learning, control theory, bandit algorithms, and machine learning.

Zenghao Chai is a PhD student at the School of Computing, National University of Singapore. He obtained his MEng from Tsinghua University and BEng from Beijing Institute of Technology. His research interests are computer vision and graphics. He has published several papers in NeurIPS, CVPR, ICCV, ECCV, ACM MM, and TMM.



Xinzhu Ma received his B.Eng and M.P's degree from Dalian University of Technology in 2017 and 2019, respectively. After that, He got the Ph.D degree from the University of Sydney in 2023. He is currently a postdoctoral researcher at the Chinese University of Hong Kong. His research interests include deep learning and computer vision.

Zhi Wang (Senior Member, IEEE) is currently an associate professor at Shenzhen International Graduate School, Tsinghua University. He received his Ph.D. in 2014 and his B.E. in 2008, both from Tsinghua University. His research areas include multimedia networks, mobile cloud computing, and large-scale machine learning systems. He was a recipient of the Natural Science Award of the Ministry of Education (First Prize) in 2017, the National Natural Science Award (Second Prize) in 2018, the Shenzhen Youth Science and Technology Award in 2019, and the Technology Invention Award of the Chinese Institute of Electronics (First Prize) in 2020. In addition, his research won the Best Paper Award of ACM Multimedia, the Best Paper Award of IEEE Transactions on Multimedia, the Outstanding Doctoral Thesis Award of China Computer Federation, the Best Student Paper Award of MMM, and the Best Paper Award of ACM Multimedia, HUMA Workshop. He is an Associate Editor of IEEE TMM and Guest Editor of ACM TIST and JCST. His research has been covered by prestigious technology media, including MIT Technology Review and Synced Review.

Wenwu Zhu (Fellow, IEEE) is currently a Professor with the Computer Science Department, Tsinghua University, Beijing, China, and the Vice Dean of National Research Center on Information Science and Technology. Prior to his current post, he was a Senior Researcher and Research Manager with Microsoft Research Asia. He was the Chief Scientist and the Director with Intel Research China, from 2004 to 2008. He was with Bell Labs New Jersey as a Member of Technical Staff during 1996–1999. His current research interests include the areas of cross-media big data and intelligence, and multimedia edge computing. He was the Editor-in-Chief for the IEEE TRANSACTIONS ON MULTIMEDIA from January 1, 2017 to December 31, 2019. He has been serving as the Chair of the steering committee for IEEE T-MM and Vice EiC for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT) since January 1, 2020. He was the recipient of the nine Best Paper Awards. He is an AAAS Fellow, SPIE Fellow and a Member of the European Academy of Sciences (Academia Europaea).