FAST ADAPTIVE TOKEN SELECTION VIA EFFICIENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision Transformers (ViTs) have achieved remarkable success in a wide range of visual tasks due to their strong global modeling capability. However, the continuous processing of full token sequences in deep Transformer layers introduces substantial inference overhead. Existing token pruning methods often struggle to balance efficiency and task adaptability: retraining-free methods rely on fixed importance heuristics, while retraining-based methods typically require additional optimization of the full model. Therefore, a key challenge is: how to learn token selection policies with both task awareness and long-term decision-making capability without updating large-scale pretrained ViTs. To address this challenge, we propose FASTER, a lightweight reinforcement learning(RL) framework for rapid task-adaptive ViT token pruning. FASTER freezes the pretrained ViT backbone and task head, and only optimizes a lightweight Actor–Critic policy network. It introduces Task-aware Sensitivity Encoding (TSE) to transform task-related knowledge embedded in the pretrained model into a reinforcement learning state representation, and learns instance-adaptive token retention policies by jointly considering token representations, spatial information, and budget constraints. Experimental results demonstrate that FASTER effectively reduces computational cost and inference latency on DeiT-B, DeiT-S, and DeiT-T, achieving a competitive trade-off between accuracy and efficiency.

1 INTRODUCTION

Vision Transformers (ViTs) (Dosovitskiy, 2020) have achieved remarkable success in various visual tasks, including image classification (Zhang et al., 2024; Sun et al., 2023; Hatamizadeh & Kautz, 2025; Bergner et al., 2025; Yang et al., 2025), object detection (Yao et al., 2023; Huang et al., 2025; Zhang et al., 2025; Hou et al., 2024; Lin et al., 2023), and image segmentation (Li et al., 2024; Wu et al., 2024; Kerssies et al., 2025; Li et al., 2025; Lee et al., 2026). However, ViTs rely on full Patch Token sequences for feature propagation across Transformer layers. As the input resolution and sequence length increase, redundant tokens continue to participate in subsequent computations, resulting in substantial computational and inference overhead.

To reduce the computational cost of ViTs, existing studies have explored various model compression techniques, including matrix factorization (Qin et al., 2023; An et al., 2025; Chang et al., 2024; Guo et al., 2024), quantization (Li et al., 2022; Huang et al., 2023; Wang et al., 2022; Li et al., 2023), knowledge distillation (Touvron et al., 2021; Chen et al., 2022; Hao et al., 2022; Chen et al., 2021), and pruning. Among them, Token pruning has attracted increasing attention by dynamically adjusting the input sequence length and selectively removing redundant tokens, allowing subsequent Transformer layers to focus on more task-relevant visual information while reducing computational cost.

Existing token pruning methods can be broadly categorized into two groups: retraining-based methods and retraining-free methods. Retraining-based approaches learn token selection or merging strategies through additional optimization, providing stronger task adaptation capabilities. For example, Wei et al. (2023) transfer information from removed tokens to retained tokens; Mao et al. (2025) jointly optimize token pruning and token merging; Lee et al. (2024) integrate multiple criteria for token fusion; and Li et al. (2026) employ PPO to jointly optimize compression ratios of tokens, multi-head self-attention (MHSA) channels, and MLP channels. Although these methods can learn

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

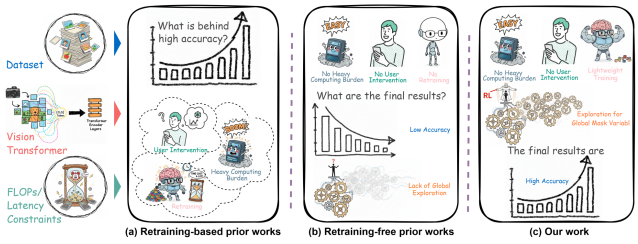


Figure 1: Comparison between (a) retraining-based token pruning methods, (b) retraining-free token pruning methods, and (c) the proposed FASTER.

more effective compression strategies for specific tasks, they typically require additional training procedures, increasing adaptation costs and limiting their deployment efficiency on large-scale pretrained ViTs.

In contrast, Retraining-free methods can be directly applied to pretrained ViTs without updating model parameters, offering lower adaptation costs and better deployment convenience. For example, Bolya et al. (2023) progressively merge redundant tokens based on token similarity; Kim et al. (2022) perform adaptive token selection according to class attention distributions; and Wang et al. (2024) rank tokens using attention maps and PageRank. However, without explicit optimization toward downstream task objectives, these methods usually rely on predefined importance metrics or heuristic rules, making it difficult to fully exploit task-related knowledge embedded in pretrained models.

Based on these observations, we revisit ViT token pruning from the perspective of **sequential decision making** and formulate it as a long-horizon decision problem. This raises an important question: **Can we learn task-aware token selection policies with low adaptation cost without updating the pretrained ViT backbone and task head, while enabling early pruning decisions to account for their long-term effects on subsequent representations and final task performance?**

To address this challenge, we propose **FASTER**, a lightweight reinforcement learning framework for rapid task-adaptive ViT token pruning. As illustrated in Figure. 1, FASTER bridges the gap between Retraining-based and Retraining-free approaches by avoiding costly fine-tuning of the entire visual model while overcoming the limitation of rule-based token selection. Specifically, FASTER introduces Task-aware Sensitivity Encoding (TSE) to transform token-level task information from the frozen ViT into a reinforcement learning (RL) state representation. By jointly considering token representations, spatial information, and budget constraints, FASTER learns adaptive token retention policies. By formulating token pruning as a **long-horizon sequential decision process**, FASTER enables pruning strategies to consider the cumulative effects of early decisions on subsequent representations and final task performance.

The main contributions of this work are summarized as follows:

- We propose FASTER, a lightweight RL framework for task-adaptive ViT Token pruning. By freezing the backbone and task head, FASTER learns dynamic Token-selection policies under different budgets.
- We introduce TSE, which injects task-related knowledge from pretrained ViTs into the RL state without imposing hard retention constraints.
- We model Token pruning as a long-horizon sequential decision process and use a task-guided Actor–Critic policy with cross-layer rewards to account for the effects of early Token decisions.
- We evaluate alternative RL strategies and demonstrate that FASTER achieves a favorable balance between computational cost and task accuracy.

2 Related Works

2.1 Token Compression for Vision Transformers

ViTs achieve global context modeling through Patch Token sequences, but repeatedly processing the full token sequence across Transformer layers introduces substantial inference overhead. To

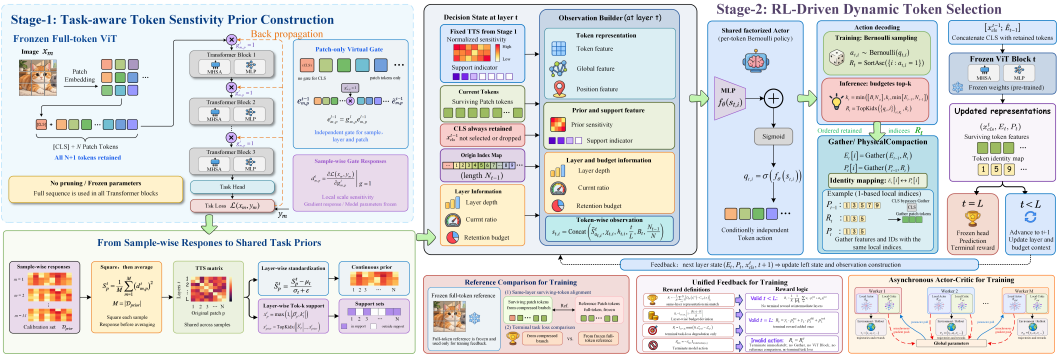


Figure 2: Overview of the FASTER framework. TSE is incorporated into the RL state, enabling the Actor–Critic policy to learn adaptive token selection under representation and budget constraints.

improve ViT efficiency, existing studies have explored token pruning and token merging approaches. Token pruning methods typically remove low-contribution tokens based on attention responses or token importance scores. For example, EViT (Liang et al., 2022), DynamicViT (Rao et al., 2021), A-ViT (Yin et al., 2022), TPS (Wei et al., 2023), and Evo-ViT (Xu et al., 2022) improve token removal from the perspectives of attention-based selection, dynamic prediction, and information preservation. Token merging methods reduce sequence length by aggregating similar tokens, such as ToMe (Bolya et al., 2023), MCTF (Lee et al., 2024), and DiffRate (Chen et al., 2023). In addition, Zero-TPrune (Wang et al., 2024) explores training-free token selection based on attention maps and graph structures. Although these methods effectively reduce the computational cost of ViTs, their token selection strategies mainly rely on local importance metrics and lack explicit task-aware modeling, making it difficult to capture the long-term impact of pruning decisions across network depth.

2.2 Reinforcement Learning for Adaptive Model Compression

RL learns decision-making policies through agent–environment interactions with the objective of maximizing long-term cumulative rewards. Recently, RL has been widely applied to model compression. For example, AMC (He et al., 2018), DRL (Chen et al., 2020), and Wang et al. (Wang & Li, 2022) employ RL to optimize CNN channel pruning strategies; TR-BERT (Ye et al., 2021) uses reinforcement learning to dynamically adjust token computation depth; G-Pruner (Yao et al., 2024), PRANCE (Li et al., 2026), and RL4EViT (Lu et al., 2025) further explore RL-based Transformer compression and cross-layer token selection. However, existing RL-based compression methods mainly focus on compression ratio search, structural optimization, or resource allocation, while token-level task relevance modeling remains insufficient. To address this limitation, FASTER introduces TSE, which incorporates task-related information from the pretrained ViT into the reinforcement learning state and leverages cross-layer long-term rewards to learn instance-adaptive token selection policies.

3 Method

3.1 Overview of FASTER

As illustrated in Figure 2, FASTER freezes the ViT backbone and task head while optimizing only a lightweight Actor–Critic policy. By incorporating task sensitivity, token representations, spatial information, and budget constraints into the RL state, FASTER learns instance-adaptive token selection through long-term rewards rather than fixed importance ranking.

3.2 Task-aware Sensitivity-guided Reinforcement Learning

Task-aware Sensitivity Encoding. FASTER computes token-level task sensitivity with a frozen ViT and incorporates it into the reinforcement learning state. Given a pretrained ViT F with L Transformer blocks, an input image x is embedded into N patch tokens of dimension D and concatenated

with a CLS token. For a sample (x_m, y_m), the task loss is denoted by $\mathcal{L}(x_m, y_m)$. Sensitivity is computed from the complete token sequence. The input of the t-th Transformer block is represented as $Z_{t-1}^{base} = [x_{\text{cls},base}^{t-1}; E_{t-1}^{base}] \in \mathbb{R}^{(N+1)\times D}$, where $base$ denotes the uncompressed reference model. The complete token sequence preserves the original patch identities and provides a unified index for sensitivity estimation across layers.

For the token representation e_p^{t-1} at original patch position p, we introduce a virtual scalar gate only for gradient computation $\tilde{e}_p^{t-1} = g_p^{t-1} e_p^{t-1}$. When $g_p^{t-1} = 1$, the forward computation remains unchanged.

The local task response of token p is defined as

$$d_{m,p}^t = \left. \frac{\partial \mathcal{L}(x_m, y_m)}{\partial g_p^{t-1}} \right|_{g=1} = \left\langle \left. \frac{\partial \mathcal{L}(x_m, y_m)}{\partial \tilde{e}_p^{t-1}} \right|_{g=1}, e_p^{t-1} \right\rangle. \tag{1}$$

This response measures the first-order influence of the token representation on the task loss. It guides policy learning but does not directly represent the loss change caused by physically removing the token.

FASTER aggregates the responses over a calibration set $\mathcal{D}_{cal}$ to obtain a sensitivity matrix $S \in \mathbb{R}^{L\times N}$, where rows correspond to Transformer blocks and columns correspond to original patch identities $S_p^t = \frac{1}{|\mathcal{D}_{cal}|} \sum_{(x_m,y_m)\in\mathcal{D}_{cal}} \left(d_{m,p}^t\right)^2$.

The sensitivity scores are normalized within each layer $\tilde{S}_p^t = \frac{S_p^t - \mu_t}{\sigma_t + \epsilon}$. Given a guidance ratio β_{guide}^t, the top-k sensitive positions form the guidance set K_{guide}^t. For a surviving token with original identity $o_{t,i}$, the binary guidance feature is defined as $\chi_{t,i} = \mathbb{I}(o_{t,i} \in K_{\text{guide}}^t)$. The continuous sensitivity representation $\tilde{S}$ and the binary feature χ serve as auxiliary observations for the policy rather than hard retention constraints. The sensitivity encoding is computed once during calibration and remains fixed during policy optimization and inference.

Markov Decision Process for Dynamic Token Selection. FASTER models progressive Token selection as a finite-horizon Markov decision process. For each image, the complete compression process forms one episode. Under a given budget schedule, the state at step t is

$$X_t = (x, y, t, B_t, x_{\text{cls}}^{t-1}, E_{t-1}, P_{t-1}), \tag{2}$$

where x and y are the input image and label used by the environment reward, while B_t is the target retention ratio relative to the original Token number N, x_{cls}^{t-1} is the current CLS representation, $E_{t-1} \in \mathbb{R}^{N_{t-1}\times D}$ is the surviving Patch-Token sequence, and P_{t-1} records the original identities of the surviving Tokens. Initially, $N_0 = N$ and $P_0 = [1, \ldots, N]$.

During policy optimization, the ViT backbone and task head remain frozen. The compressed and reference paths share the same model parameters. The reference path is used only to compute training rewards, while the policy observation uses only information available during inference.

At step t, the action is a binary retention decision. Here, $a_{t,i} = 1$ retains the i-th surviving Token and $a_{t,i} = 0$ removes it. The selected Tokens are gathered and passed to the next Transformer block. An all-zero action is treated as invalid, terminates the episode, and receives a penalty. After the final-layer decision, the policy receives an additional terminal task reward.

Task-guided Token Observation and Factorized Policy. FASTER constructs the Actor observation from task guidance, Token representations, spatial information, and budget states. For the i-th surviving Token at layer t, its original identity is $o_{t,i} = P_{t-1}[i]$. The global context of the surviving Token set is $\bar{e}_{t-1} = \frac{1}{N_{t-1}} \sum_{j=1}^{N_{t-1}} e_j^{t-1}$. The global, local, and positional features are projected and concatenated as

$$h_{t,i} = \text{Concat}(\phi_{\text{glob}}(\bar{e}_{t-1}), \phi_{\text{loc}}(e_i^{t-1}), \phi_{\text{pos}}(\text{Pos}(o_{t,i}))). \tag{3}$$

The projection modules are optimized jointly with the Actor. Combining these features with the sensitivity guidance and pruning state gives

$$s_{t,i} = \text{Concat}(\tilde{S}_{o_{t,i}}^t, \chi_{t,i}, h_{t,i}, \frac{t}{L}, B_t, \frac{N_{t-1}}{N}). \tag{4}$$

where $\tilde{S}$ and χ denote the continuous sensitivity representation and the discrete task guidance feature, respectively. They guide policy learning but do not impose a fixed Token ranking.

FASTER uses a parameter-shared factorized Bernoulli policy $\pi_\theta(a_t|X_t) = \prod_{i=1}^{N_{t-1}} \pi_\theta(a_{t,i}|s_{t,i})$. For each Token, the retention probability is computed as $l_{t,i} = f_\theta(s_{t,i}), q_{t,i} = \sigma(l_{t,i})$. Because task sensitivity is included in the observation, no additional handcrafted sensitivity bias is required. Token selection is determined jointly by task guidance, instance-specific features, and long-term rewards. During training, actions are sampled as $a_{t,i} \sim Bernoulli(q_{t,i})$, to explore different Token subsets. During inference, the learned probabilities are used for deterministic budget-constrained Top-k selection.

3.3 Budget-constrained Token Decoding and Physical Compression.

FASTER converts the learned retention policy into computational reduction through budget-constrained decoding. The budget determines the number of retained Tokens, while the Actor probabilities determine which Tokens are selected. Given the monotonically decreasing schedule $1 \geq B_1 \geq \cdots \geq B_L > 0$, where B_t is the target retention ratio at block t relative to the original Token number, the target and feasible retention sizes are $\bar{k}_t = \max\{1, \lfloor B_t N \rfloor\}$, and $k_t = \min\{\bar{k}_t, N_{t-1}\}$. Defining the budget relative to the original Token number avoids cumulative deviation during progressive compression.

Stochastic Exploration and Deterministic Decoding. During training, FASTER samples Token actions from the Actor probabilities: $a_{t,i} \sim Bernoulli(q_{t,i})$, with selected indices and Token count given by $R_t = SortAsc(\{i : a_{t,i} = 1\}), N_t = \sum_i a_{t,i}$. The sampled actions are not projected onto a fixed budget, allowing the policy to learn the long-term effects of different Token combinations. When $N_t = 0$, the action is treated as invalid.

During inference, FASTER performs deterministic budget-constrained decoding. Given k_t, it selects $\tilde{R}_t = TopKIdx(\{q_{t,i}\}_{i=1}^{N_{t-1}}, k_t)$, and sorts the selected indices to preserve their original order: $R_t = SortAsc(\tilde{R}_t)$. Under the fixed cardinality constraint, this Top-k rule selects the maximum-probability action of the factorized Bernoulli policy.

Physical Compression and Identity Preservation. Given the selected token set R_t, FASTER compresses the token sequence and updates the identity mapping simultaneously $\hat{E}_{t-1} = Gather(E_{t-1}, R_t) \in \mathbb{R}^{N_t \times D}, P_t = Gather(P_{t-1}, R_t)$. The compressed patch tokens are concatenated with the CLS token and processed by the t-th Transformer block $[x^t_{\text{cls}}, E_t] = Block_t([x^{t-1}_{\text{cls}}, \hat{E}_{t-1}])$. For checkpoints with absolute positional embeddings, these embeddings are added before the first pruning step and remain encoded in the gathered representations. The identity map P_t is additionally used for position-dependent policy features and reference lookup. Thus, the block processes only $N_t + 1$ tokens, and discarded Tokens do not participate in subsequent computation. By maintaining $E_t[i] \leftrightarrow P_t[i]$, FASTER preserves the correspondence between each surviving Token and its original Patch identity, enabling sensitivity lookup, positional feature extraction, and reference-feature alignment.

Structural Guarantees. Under a monotonically non-increasing budget schedule, FASTER's Top-k selection controls the number of Tokens at each layer and keeps the sequence nonempty. By synchronously updating the Token features and identity mapping, physical compaction preserves the correspondence between each surviving Token and its original Patch. The formal proof is provided in Appendix B. During deployment, only the frozen ViT, fixed sensitivity encoding, Actor network, and Token-compression path are retained. The Critic network, reference branch, and sensitivity-gradient computation are removed. During training, the frozen ViT and discrete Token selection form the reinforcement learning environment, and the Actor is optimized from sampled actions and accumulated rewards without backpropagating through discrete Token removal.

3.4 Reward Design for Long-term Token Selection

FASTER uses representation consistency, budget consistency, and task-level feedback to connect local Token decisions with long-term compression quality. The ViT backbone, task head, and reference path remain frozen, and the representation and task losses are clipped for stable optimization.

Representation Consistency Feedback. FASTER uses the identity mapping to align the compressed representation E_t with the corresponding features from the complete reference path

$$D_{\text{rec}}^t = \frac{1}{N_t D} \sum_{i=1}^{N_t} \left\| E_t[i] - sg(E_t^{base}[P_t[i]]) \right\|_2^2 . \tag{5}$$

where $sg(\cdot)$ denotes stop-gradient. The representation reward is $R_{\text{rec}}^t = -\min(D_{\text{rec}}^t, C_{\text{rec}})$. It provides intermediate representation-preservation feedback, while final task performance is evaluated by the terminal reward.

Budget Consistency Feedback. To constrain stochastic exploration under the prescribed computational budget, FASTER defines $R_{\text{bud}}^t = -w_b((N_t - \bar{k}_t)/N)^2, w_b > 0$. where $\bar{k}_t$ is defined relative to the original Token number, preventing budget deviations from accumulating during progressive compression. This reward encourages the desired computational scale and discourages excessive early Token removal. The expectation decomposition of the budget deviation under Bernoulli policy sampling, including its variance term, is provided in Appendix C.

Intermediate and Terminal Feedback. For valid actions, the intermediate reward combines representation and budget feedback $r_t^{\text{step}} = w_r R_{\text{rec}}^t + R_{\text{bud}}^t, w_r > 0$. For an episode that reaches the final Transformer layer, FASTER evaluates the task degradation between the compressed and reference paths. Let $\mathcal{L}_{\text{pruned}}$ and $\mathcal{L}_{\text{base}}$ denote their respective task losses $D_{\text{task}} = \max(0, \mathcal{L}_{\text{pruned}} - sg(\mathcal{L}_{\text{base}}))$, and $R_{\text{term}} = -w_{\text{task}} \min(D_{\text{task}}, C_{\text{task}})$. This terminal feedback penalizes accumulated task degradation without imposing a hard performance constraint.

Unified Feedback and Invalid Actions. FASTER uses an undiscounted finite-horizon return with $\gamma = 1$. To prevent the policy from removing all Tokens to terminate an episode early, the invalid-action penalty at layer t is defined as $r_{\text{invalid}}(t) = -(L - t + 1)C_{\text{step}} - C_{\text{term}} - \eta, \eta > 0$. The complete environmental feedback is

$$r_t = \begin{cases} r_{\text{invalid}}(t), & N_t = 0, \\ r_t^{\text{step}}, & N_t > 0,\ t < L, \\ r_t^{\text{step}} + R_{\text{term}}, & N_t > 0,\ t = L. \end{cases} \tag{6}$$

3.5 Lightweight Actor–Critic Optimization

FASTER uses asynchronous Advantage Actor–Critic (A3C) Mnih et al. (2016b) to optimize the Token-retention policy from long-term feedback across Transformer layers. For the finite-horizon process, we set $\gamma = 1$. If an episode terminates at $\tau \le L$, the objective is $J(\theta) = \mathbb{E}\left[\sum_{t=1}^{\tau} r_t\right]$. This objective jointly exploits task sensitivity, instance-specific representations, and previous decisions. For worker w, local Actor and Critic parameters collect trajectories of length H, terminated either by the episode or by the rollout limit.

Task-aware Critic Representation and Value Estimation. For the i-th surviving token at layer t, FASTER constructs the Critic representation $z_{t,i} = \text{Concat}(\tilde{S}_{o_{t,i}}^t, \chi_{t,i}, sg(h_{t,i}))$, where $\tilde{S}$ and χ denote the continuous sensitivity representation and discrete task guidance feature. A shared value encoder produces $u_{t,i} = \psi_\omega(z_{t,i})$. The set-level representation is $\bar{u}_t = \frac{1}{N_{t-1}} \sum_{i=1}^{N_{t-1}} u_{t,i}$, which is combined with the layer index and budget state $c_t = \text{Concat}(\bar{u}_t, t/L, B_t, N_{t-1}/N)$. The Critic $V_\omega(c_t)$ estimates the expected cumulative return and serves as the baseline for joint Token decisions.

Multi-step Returns and Layer-level Advantages. For a truncated trajectory, the future value is estimated by

$$\hat{V}_{t+H} = \begin{cases} 0, & \text{episode terminated}, \\ sg(V_{\omega^{(w)}}(c_{t+H})), & \text{otherwise}. \end{cases} \tag{7}$$

The multi-step return is $G_{t+j} = \sum_{l=0}^{H-j-1} r_{t+j+l} + \hat{V}_{t+H}, j = 0, \ldots, H-1$. Successfully completed episodes and invalid actions are both treated as terminal states, so no additional terminal value is used. The layer-level advantage is $A_{t+j} = G_{t+j} - V_{\omega^{(w)}}(c_{t+j})$.

Actor–Critic Updates. The Actor is optimized using the joint action probability

$$\log \pi_{\theta^{(w)}}(a_t | X_t) = \sum_{i=1}^{N_{t-1}} [a_{t,i} \log q_{t,i}^{(w)} + (1 - a_{t,i}) \log(1 - q_{t,i}^{(w)})]. \tag{8}$$

To encourage early exploration, FASTER uses the average Token entropy

$$\bar{H}_t^{(w)} = \frac{1}{N_{t-1}} \sum_{i=1}^{N_{t-1}} H(q_{t,i}^{(w)}), \tag{9}$$

where $H(q) = -q \log q - (1-q)\log(1-q)$. The Actor loss is

$$\mathcal{L}_{actor}^{(w)} = -\sum_{j=0}^{H-1} \left[sg(A_{t+j}) \log \pi_{\theta^{(w)}}(a_{t+j}|X_{t+j}) + \lambda_{ent}(u)\bar{H}_{t+j}^{(w)} \right]. \tag{10}$$

The Critic is trained using the multi-step return

$$\mathcal{L}_{critic}^{(w)} = \frac{1}{2} \sum_{j=0}^{H-1} [sg(G_{t+j}) - V_{\omega^{(w)}}(c_{t+j})]^2. \tag{11}$$

The combined objective is $\mathcal{L}^{(w)} = \mathcal{L}_{actor}^{(w)} + \lambda_v \mathcal{L}_{critic}^{(w)}$. Stop-gradient operations isolate the Actor, Critic, and frozen visual model. Only the lightweight Actor–Critic parameters are updated during training, while the ViT backbone, task head, and sensitivity encoding remain frozen. At deployment, FASTER retains only the frozen ViT, fixed sensitivity encoding, Actor, and budget-constrained compression path, removing the Critic, reference branch, and sensitivity-gradient computation.

4 Experiments

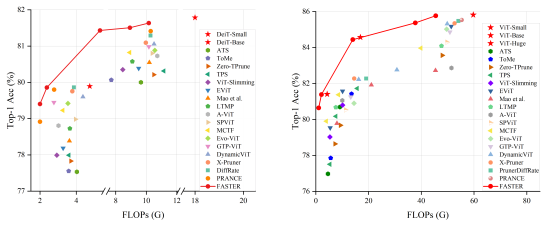
4.1 Experimental Setup

We evaluate FASTER on four ViT backbones, namely ViT-L, DeiT-B, DeiT-S, and DeiT-T, using timm Wightman et al. (2019) and ImageNet-1K Deng et al. (2009). All experiments run on a single NVIDIA A100 GPU. Retraining-based methods are evaluated over five independent runs, with mean and standard deviation reported, whereas retraining-free methods are evaluated once deterministically. Details are provided in Appendix D.

Table 1: Accuracy and efficiency comparison for DeiT-T/S/B under different computational budgets.

	DeiT-T under the 0.6G Budget					DeiT-S under the 2.0G Budget					DeiT-B under the 15.0G Budget				
Method	**GFLOPs ↓**	**Top-1 Acc. (%) ↑**	**Latency (ms) ↓**	**Throughput (img/s) ↑**	**Policy Time (GPU h) ↓**	**GFLOPs ↓**	**Top-1 Acc. (%) ↑**	**Latency (ms) ↓**	**Throughput (img/s) ↑**	**Policy Time (GPU h) ↓**	**GFLOPs ↓**	**Top-1 Acc. (%) ↑**	**Latency (ms) ↓**	**Throughput (img/s) ↑**	**Policy Time (GPU h) ↓**
Baseline	1.3	72.20	2.15 ± 0.04	465.1	–	4.6	79.89	5.20	192.3	–	17.6	81.78	9.80 ± 0.12	102.0	–
+ATS Kim et al. (2022)	0.6	69.66	1.75	571.4	–	2.0	76.64	4.24	235.8	–	15.0	80.45	9.30	107.5	–
+ToMe Bolya et al. (2023)	0.6	69.59	1.73	578.0	–	2.0	76.59	4.18	239.2	–	15.0	80.53	9.18	108.9	–
+Zero-TPrune Wang et al. (2024)	0.6	69.89	1.70	588.2	–	2.0	77.23	4.12	242.7	–	15.0	80.57	9.12	109.6	–
+TPS Wei et al. (2023)	0.6	70.10 ± 0.05	1.67 ± 0.04	598.8	0.4	2.0	77.50 ± 0.06	4.08 ± 0.07	245.1	1.3	15.0	80.64 ± 0.03	8.95 ± 0.11	111.7	4.8
+ViT-Slimming Chavan et al. (2022)	0.6	70.17 ± 0.05	1.66 ± 0.05	602.4	1.2	2.0	77.60 ± 0.05	4.04 ± 0.10	247.5	4.2	15.0	80.74 ± 0.03	8.88 ± 0.11	112.6	16.0
+EViT Liang et al. (2022)	0.6	70.26 ± 0.05	1.64 ± 0.04	609.8	0.4	2.0	77.74 ± 0.05	4.00 ± 0.08	250.0	1.5	15.0	80.70 ± 0.03	8.82 ± 0.11	113.4	5.6
+Mao et al. Mao et al. (2025)	0.6	70.34 ± 0.06	1.62 ± 0.04	617.3	0.6	2.0	77.85 ± 0.06	3.96 ± 0.07	252.5	2.0	15.0	80.79 ± 0.04	8.76 ± 0.10	114.2	7.5
+LTMP Bonnaerens & Dambre (2023)	0.6	70.54 ± 0.04	1.59 ± 0.04	628.9	0.4	2.0	78.17 ± 0.04	3.84 ± 0.08	260.4	1.4	15.0	80.86 ± 0.03	8.65 ± 0.10	115.6	5.2
+A-ViT Yin et al. (2022)	0.6	70.73 ± 0.05	1.61 ± 0.04	621.1	0.4	2.0	78.55 ± 0.05	3.88 ± 0.08	257.7	1.6	15.0	80.94 ± 0.03	8.72 ± 0.11	114.7	6.0
+SPViT Kong et al. (2022)	0.6	70.67 ± 0.05	1.61 ± 0.04	621.1	0.7	2.0	78.43 ± 0.05	3.92 ± 0.09	255.1	2.5	15.0	80.96 ± 0.03	8.79 ± 0.11	113.8	9.6
+MCTF Lee et al. (2024)	0.6	70.94 ± 0.04	1.57 ± 0.04	636.9	0.6	2.0	78.87 ± 0.04	3.80 ± 0.08	263.2	2.1	15.0	81.00 ± 0.02	8.58 ± 0.10	116.6	7.9
+Evo-ViT Xu et al. (2022)	0.6	71.09 ± 0.04	1.56 ± 0.04	641.0	0.5	2.0	79.10 ± 0.04	3.76 ± 0.08	266.0	1.7	15.0	81.02 ± 0.03	8.62 ± 0.10	116.0	6.4
+GTP-ViT Xu et al. (2024)	0.6	71.22 ± 0.04	1.54 ± 0.04	649.4	0.6	2.0	79.28 ± 0.03	3.72 ± 0.07	268.8	2.2	15.0	81.12 ± 0.02	8.50 ± 0.10	117.6	8.4
+DynamicViT Rao et al. (2021)	0.6	71.15 ± 0.04	1.52 ± 0.04	657.9	0.4	2.0	79.04 ± 0.04	3.68 ± 0.07	271.7	1.4	15.0	81.14 ± 0.02	8.44 ± 0.09	118.5	5.2
+X-Pruner Yu & Xiang (2023)	0.6	71.34 ± 0.03	1.50 ± 0.04	666.7	0.8	2.0	79.34 ± 0.03	3.62 ± 0.08	276.2	2.7	15.0	81.24 ± 0.02	8.38 ± 0.09	119.3	10.4
+DiffRate Chen et al. (2023)	0.6	71.49 ± 0.03	1.47 ± 0.04	680.3	0.5	2.0	79.36 ± 0.03	3.55 ± 0.08	281.7	1.9	15.0	81.37 ± 0.02	8.31 ± 0.09	120.3	7.2
+PRANCE Li et al. (2026)	0.6	71.58 ± 0.04	1.44 ± 0.04	694.4	0.3	2.0	78.91 ± 0.04	3.45 ± 0.08	289.9	1.0	15.0	81.50 ± 0.02	8.26 ± 0.09	121.1	4.0
+FASTER (Ours)	0.6	**71.85 ± 0.03**	**1.41 ± 0.04**	**720.2**	**0.1**	2.0	**79.42 ± 0.03**	**3.30 ± 0.08**	**305.0**	**0.3**	15.0	**81.83 ± 0.02**	**8.20 ± 0.08**	**122.5**	**1.0**

4.2 Comparison with State-of-the-Art Methods



(a) (b)

Figure 3: Top-1 accuracy versus GFLOPs on DeiT-S/B and ViT-S/B/H backbones.

Accuracy and computational cost. Figure 3 compares Top-1 accuracy and GFLOPs across DeiT and ViT backbones. FASTER generally achieves higher accuracy at comparable computational cost. At 0.6G, 2.0G, and 15.0G, it reaches 71.85%, 79.42%, and 81.83% Top-1 accuracy on DeiT-T, DeiT-S, and DeiT-B, respectively (Table 1). Task-aware sensitivity encoding and cross-layer rewards guide the policy to retain task-relevant Tokens across model scales and budgets.

Inference latency and throughput. At the same budgets, FASTER achieves end-to-end

latencies of 1.41, 3.30, and 8.20 ms, with throughputs of 720.2, 305.0, and 122.5 images/s on DeiT-T, DeiT-S, and DeiT-B, respectively (Table 1). Latency includes the compressed backbone, Actor inference, Token selection, and physical gather.

Policy adaptation cost. Unlike training-free methods, FASTER learns a policy while keeping the ViT backbone and task head frozen. Optimizing only the lightweight Actor–Critic network takes 0.1, 0.3, and 1.0 GPU hours on DeiT-T, DeiT-S, and DeiT-B, respectively. TSE supplies task-relevant guidance to the policy state, helping reduce unproductive exploration.

4.3 Effect of Different RL Optimizers on FASTER

Table 2: Comparison of RL optimizers under different backbones and computational budgets.

Backbone	GFLOPs	Top-1 Accuracy (%)							
		Random	Q-Learning	SARSA	REINFORCE	A2C	TRPO	PPO	A3C
DeiT-T	0.6	65.80 ± 0.18	67.12 ± 0.15	66.94 ± 0.17	69.35 ± 0.12	70.91 ± 0.08	71.10 ± 0.07	71.53 ± 0.05	**71.85 ± 0.03**
DeiT-S	2.0	73.40 ± 0.21	74.06 ± 0.16	73.88 ± 0.18	76.21 ± 0.11	78.62 ± 0.07	78.90 ± 0.08	79.14 ± 0.05	**79.42 ± 0.03**
	3.0	75.91 ± 0.17	76.42 ± 0.14	76.60 ± 0.13	78.15 ± 0.10	79.30 ± 0.06	79.58 ± 0.05	79.50 ± 0.06	**79.67 ± 0.04**
DeiT-B	10.0	76.13 ± 0.19	76.62 ± 0.17	76.44 ± 0.15	78.60 ± 0.12	80.51 ± 0.08	81.04 ± 0.06	81.22 ± 0.04	**81.55 ± 0.03**
	15.0	78.52 ± 0.16	78.97 ± 0.12	79.08 ± 0.14	80.32 ± 0.09	81.08 ± 0.06	81.30 ± 0.05	81.56 ± 0.04	**81.83 ± 0.02**

FASTER models layer-wise Token retention with factorized Bernoulli actions. A3C Mnih et al. (2016b) directly optimizes this stochastic policy and combines a Critic baseline, multi-step returns, and asynchronous trajectory collection to handle delayed rewards. We compare it with Q-learning Watkins & Dayan (1992), SARSA Sutton et al. (1998), REINFORCE Williams (1992), A2C Mnih et al. (2016a), TRPO Schulman et al. (2015), and PPO Schulman et al. (2017) under matched training and evaluation settings, using token-wise keep/drop values for the value-based methods and factorized Bernoulli actions for the policy-based methods. As shown in Table 2, A3C achieves the highest accuracy in all tested settings, reaching 71.85%, 79.42%, and 81.83% on DeiT-T, DeiT-S, and DeiT-B at the main budgets. We therefore use A3C as FASTER's default optimizer; further analysis is provided in Appendix F.

4.4 Ablation Studies

Table 3: Component ablation of FASTER.

Method	TSE	RL	ID	Reward	Comp.	GFLOPs	Top-1 (%)	Latency (ms)	Throughput (img/s)	Speedup
Baseline	×	×	×	×	×	17.6	81.78	9.80	102.0	1.00×
Random Selection	×	×	×	×	✓	15.0	79.12 ± 0.12	8.94	111.7	1.10×
w/o Full TSE Module	×	✓	✓	✓	✓	15.0	81.02 ± 0.05	8.31	120.3	1.18×
w/o RL	✓	×	✓	✓	✓	15.0	81.28 ± 0.04	8.27	120.9	1.19×
w/o Identity Mapping	✓	✓	×	✓	✓	15.0	81.50 ± 0.03	8.24	121.4	1.19×
w/o Reward Components	✓	✓	✓	×	✓	15.0	81.60 ± 0.03	8.23	121.5	1.19×
w/o Physical Compaction	✓	✓	✓	✓	×	15.0	81.72 ± 0.02	9.05	110.8	1.10×
FASTER (Ours)	✓	✓	✓	✓	✓	**15.0**	**81.83 ± 0.02**	**8.20**	**122.5**	**1.20×**

Component Ablation. As shown in Table 3, on DeiT-B at 15 GFLOPs, random token selection achieves only 79.12% Top-1 accuracy. Removing the full TSE module or the learned policy lowers accuracy to 81.02% and 81.28%; removing identity mapping or reward components also degrades performance. Together, these components enable FASTER to reach 81.83% accuracy with an end-to-end latency of 8.20 ms. The fine-grained state ablation in Appendix G removes only TSE features from the policy observation and is therefore distinct from this full-module ablation.

Table 4: Ablation of execution mechanisms.

Backbone	Execution	Nominal GFLOPs	Top-1 (%)	Latency (ms)	Throughput (img/s)	Speedup
DeiT-T	Dense Baseline	1.3	72.20	2.15	465.1	1.00×
	Zero Masking	0.6	71.84 ± 0.03	2.03	492.6	1.06×
	Padded Logical Removal	0.6	71.84 ± 0.03	1.78	561.8	1.17×
	Physical Gather	**0.6**	**71.85 ± 0.03**	**1.41**	**720.2**	**1.55×**
DeiT-S	Dense Baseline	4.6	79.89	5.20	192.3	1.00×
	Zero Masking	2.0	79.40 ± 0.03	4.96	201.6	1.05×
	Padded Logical Removal	2.0	79.40 ± 0.03	4.21	237.5	1.27×
	Physical Gather	**2.0**	**79.42 ± 0.03**	**3.30**	**305.0**	**1.58×**
DeiT-B	Dense Baseline	17.6	81.78	9.80	102.0	1.00×
	Zero Masking	15.0	81.81 ± 0.02	9.52	105.0	1.03×
	Padded Logical Removal	15.0	81.81 ± 0.02	8.93	112.0	1.06×
	Physical Gather	**15.0**	**81.83 ± 0.02**	**8.20**	**122.5**	**1.20×**

Token Execution Mechanisms Ablation. Table 4 compares execution mechanisms using the same selected Token subsets. Unlike Zero Masking and Padded Logical Removal, Physical Gather reduces the sequence length processed by subsequent layers. It achieves latencies of 1.41, 3.30, and 8.20 ms on DeiT-T, DeiT-S, and DeiT-B, corresponding to 1.55×, 1.58×, and 1.20× speedups over the uncompressed baselines while maintaining comparable accuracy.

4.5 Visualization of the Pruning Process

Figure 4 maps the Tokens retained after successive Transformer blocks to their original image positions, with white regions indicating pruned Tokens. Guided by TSE and the RL policy, FASTER progressively removes less task-relevant regions while adapting the retained positions to each image.

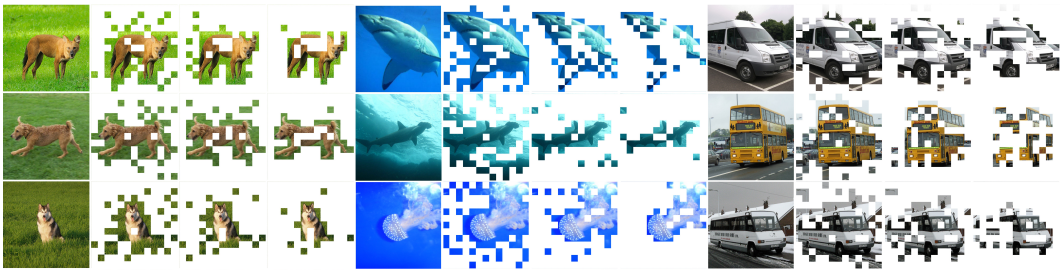


Figure 4: Layer-wise token pruning visualization of FASTER on ImageNet-1K.

The identity mapping preserves original Patch coordinates through physical compaction, enabling accurate visualization and subsequent sensitivity lookup.

Table 5: Downstream classification accuracy (%) of DeiT-B with frozen ImageNet-trained Token-selection policies.

Dataset	DeiT-B	ATS	ToMe	Zero-TPrune	TPS	ViT-Slimming	EViT	Mao et al.	LTMP	A-ViT	SPViT	MCTF	Evo-ViT	GTP-ViT	DynamicViT	X-Pruner	DiffRate	PRANCE	FASTER
Flowers-102	98.7	96.1	96.3	96.8	97.0	97.0	97.1	97.1	97.2	97.1	97.1	97.2	97.2	97.3	97.3	97.3	97.3	97.4	**97.6**
Oxford-IIIT Pets	91.4	89.2	88.3	88.5	89.4	89.3	89.5	89.6	89.6	89.6	89.5	89.9	89.8	89.8	89.7	89.9	89.9	89.8	**90.1**
DTD	75.5	72.3	72.0	72.8	73.0	72.9	73.1	73.1	73.2	73.2	73.1	73.3	73.2	73.3	73.2	73.4	73.4	73.4	**73.7**
MIT Indoor-67	78.4	75.2	75.2	76.0	76.2	76.1	76.4	76.4	76.4	76.4	76.3	76.3	76.5	76.5	76.5	76.6	76.6	76.6	**76.8**
CUB-200	79.7	76.0	76.3	76.4	76.8	76.7	77.0	77.0	77.1	77.0	77.0	77.2	77.2	77.3	77.1	77.3	77.3	77.4	**77.8**
FGVC-Aircraft	80.5	78.2	78.1	77.8	78.6	78.5	78.8	78.9	78.9	78.9	78.8	78.9	79.0	79.1	79.1	79.2	79.2	79.3	**79.7**
Stanford Cars	92.5	88.7	88.5	89.3	89.8	89.7	90.0	90.0	90.1	90.1	90.0	90.2	90.2	90.3	90.3	90.3	90.4	90.4	**90.7**

Table 6: Downstream semantic segmentation performance of DeiT-B with a frozen ImageNet-trained Token-selection policy on Pascal VOC and COCO.

Metric	DeiT-B	ATS	ToMe	Zero-TPrune	TPS	ViT-Slimming	EViT	Mao et al.	LTMP	A-ViT	SPViT	MCTF	Evo-ViT	GTP-ViT	DynamicViT	X-Pruner	DiffRate	PRANCE	FASTER
Pascal VOC mIoU (%)	79.5	77.8	78.0	78.2	78.4	78.3	78.5	77.8	78.4	78.3	78.2	78.1	78.5	78.6	78.6	78.7	78.7	78.8	**79.3**
COCO mIoU (%)	59.9	58.2	58.4	58.6	58.8	58.6	58.9	58.2	58.7	58.6	58.5	58.7	58.9	58.9	59.0	59.0	59.0	59.1	**59.6**
GFLOPs	17.6	10.3	10.2	10.1	10.2	10.4	10.0	10.7	10.4	10.5	10.6	10.5	10.1	10.0	10.0	9.9	9.9	9.9	**9.5**

4.6 Downstream Transferability

Downstream classification. As shown in Table 5, FASTER achieves the highest accuracy among the compressed methods on all seven datasets, reaching 97.6%, 90.1%, and 73.7% on Flowers-102, Oxford-IIIT Pets, and DTD, and 76.8%, 77.8%, 79.7%, and 90.7% on MIT Indoor-67, CUB-200, FGVC-Aircraft, and Stanford Cars. These results across fine-grained, texture-based, and scene recognition tasks demonstrate that the learned Token-selection policy preserves transferable visual information without task-specific policy retraining. Only the task-specific prediction head is trained for downstream evaluation; the Token-selection policy remains frozen.

Downstream segmentation. As shown in Table 6, FASTER achieves 79.3% and 59.6% mIoU on Pascal VOC and COCO at 9.5 GFLOPs, outperforming PRANCE by 0.5 percentage points on both benchmarks. Compared with the uncompressed DeiT-B, it reduces computation by approximately 46% while maintaining competitive segmentation performance. This result shows that FASTER preserves spatially informative features and transfers beyond image-level classification to dense prediction tasks.

5 Conclusion

We propose FASTER, a lightweight RL framework for task-adaptive dynamic Token pruning in ViT. FASTER freezes the pretrained ViT backbone and task head, optimizes only a lightweight Actor–Critic policy, and incorporates TSE into the RL state. With a factorized Bernoulli policy, FASTER formulates Token pruning as a layer-wise long-horizon decision process, allowing the policy to retain task-relevant Tokens according to the input content while accounting for the effects of early decisions on subsequent representations and final task performance. Experiments on DeiT-B, DeiT-S, and DeiT-T show that FASTER maintains high classification accuracy while reducing computational cost, policy adaptation cost, and end-to-end inference latency, and improving throughput. Downstream classification and semantic segmentation experiments further demonstrate that the learned Token-selection policy preserves transferable visual representations and remains effective across vision tasks beyond ImageNet classification.

AI USE STATEMENT

In this work, generative AI tools were used only to assist with language polishing, Chinese–English translation, LaTeX formatting, and the organization of the manuscript structure and presentation. Generative AI tools were not used to generate experimental data, conduct experiments, perform data analysis, design the research methodology, or draw scientific conclusions. All AI-assisted content was reviewed, revised, and verified by the authors. The experimental results, theoretical derivations, method design, and final conclusions were independently developed and verified by the authors. The authors take full responsibility for the entire content of this manuscript, including the text and formatting prepared with the assistance of generative AI tools.

ETHICS STATEMENT

This work does not involve human participants, animal experiments, personal privacy data, or sensitive personal information. The ImageNet-1K dataset and the downstream public datasets used in this study are subject to their respective public release licenses and usage policies. This work focuses on Token pruning and inference-efficiency optimization for Vision Transformers and does not involve biomedical intervention, individual identification, or other high-risk applications. The authors declare no undisclosed conflicts of interest.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed descriptions of FASTER's task-aware sensitivity encoding, reinforcement-learning state and action spaces, reward function, Token-budget schedule, physical Token compaction, and identity-mapping mechanism. All experiments use specified backbones, computational budgets, training protocols, and evaluation metrics. For methods requiring training, results are reported using the mean and standard deviation over multiple independent runs. The main paper and appendix describe the experimental settings, ablation studies, downstream evaluations, and formal proof of structural correctness. The experiments use the public ImageNet-1K and downstream datasets, and the model-training and inference configurations follow the settings described in the manuscript. The code, configuration files, and relevant experimental scripts will be provided in the supplementary materials to facilitate reproduction of the reported results.

REFERENCES

Sangho An, Jinwoo Kim, Keonho Lee, Jingang Huh, Chanwoong Kwak, Yujin Lee, Moonsub Jin, and Jangho Kim. Sparse structure exploration and re-optimization for vision transformer. In *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pp. 111–131. PMLR, 2025.

Benjamin Bergner, Christoph Lippert, and Aravindh Mahendran. Token cropr: Faster ViTs for quite a few tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9740–9750, 2025. URL https://github.com/benbergner/cropr.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023.

Maxim Bonnaerens and Joni Dambre. Learned thresholds token merging and pruning for vision transformers. *Transactions on Machine Learning Research*, 2023.

Chi-Chih Chang, Yuan-Yao Sung, Shixing Yu, Ning-Chi Huang, Diana Marculescu, and Kai-Chiang Wu. Flora: Fine-grained low-rank architecture search for vision transformer. In *WACV*, pp. 2482–2491, 2024.

Arnav Chavan, Zhiqiang Shen, Zhuang Liu, Zechun Liu, Kwang-Ting Cheng, and Eric 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*, pp. 4931–4941, 2022.

Jianda Chen, Shangyu Chen, and Sinno Jialin Pan. Storage efficient and dynamic flexible runtime channel pruning via deep reinforcement learning. *NeurIPS*, 33:14747–14758, 2020.

Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Fei Chao, Rongrong Ji, Yu Qiao, and Ping Luo. DiffRate: Differentiable compression rate for efficient vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17164–17174, 2023.

Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *CVPR*, pp. 5008–5017, 2021.

Xianing Chen, Qiong Cao, Yujie Zhong, Jing Zhang, Shenghua Gao, and Dacheng Tao. Dearkd: data-efficient early knowledge distillation for vision transformers. In *CVPR*, pp. 12052–12062, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Yangyang Guo, Guangzhi Wang, and Mohan Kankanhalli. Pela: Learning parameter-efficient models with low-rank approximation. In *CVPR*, pp. 15699–15709, 2024.

Zhiwei Hao, Jianyuan Guo, Ding Jia, Kai Han, Yehui Tang, Chao Zhang, Han Hu, and Yunhe Wang. Learning efficient vision transformers via fine-grained manifold distillation. *NeurIPS*, 35:9164–9175, 2022.

Ali Hatamizadeh and Jan Kautz. MambaVision: A hybrid Mamba-transformer vision backbone. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 25261–25270, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Hatamizadeh_MambaVision_A_Hybrid_Mamba-Transformer_Vision_Backbone_CVPR_2025_paper.html.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *ECCV*, pp. 784–800, 2018.

Xiuquan Hou, Meiqin Liu, Senlin Zhang, Ping Wei, and Badong Chen. Salience DETR: Enhancing detection transformer with hierarchical salience filtering refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17574–17583, 2024. doi: 10.1109/CVPR52733.2024.01664.

Shihua Huang, Zhichao Lu, Xiaodong Cun, Yongjun Yu, Xiao Zhou, and Xi Shen. DEIM: DETR with improved matching for fast convergence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15162–15171, 2025. doi: 10.1109/CVPR52734.2025.01412.

Xijie Huang, Zhiqiang Shen, Pingcheng Dong, and Kwang-Ting Cheng. Quantization variation: A new perspective on training transformers with low-bit precision. *TMLR*, 2023.

Tommie Kerssies, Niccolò Cavagnero, Alexander Hermans, Narges Norouzi, Giuseppe Averta, Bastian Leibe, Gijs Dubbelman, and Daan de Geus. Your ViT is secretly an image segmentation model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Kerssies_Your_ViT_is_Secretly_an_Image_Segmentation_Model_CVPR_2025_paper.html.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. Learned token pruning for transformers. In *KDD*, pp. 784–794, 2022.

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, Minghai Qin, and Yanzhi Wang. SPViT: Enabling faster vision transformers via latency-aware soft token pruning. In *European Conference on Computer Vision*, pp. 620–640. Springer, 2022.

ByeongCheol Lee, Hyun Seok Seong, Sangeek Hyun, Gilhan Park, WonJun Moon, and Jae-Pil Heo. Looking beyond the window: Global-local aligned CLIP for training-free open-vocabulary semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Lee_Looking_Beyond_the_Window_Global-Local_Aligned_CLIP_for_Training-free_Open-Vocabulary_CVPR_2026_paper.html.

Sanghyeok Lee, Joonmyung Choi, and Hyunwoo J Kim. Multi-criteria token fusion with one-step-ahead attention for efficient vision transformers. In *CVPR*, pp. 15741–15750, 2024.

Kaiyu Li, Ruixun Liu, Xiangyong Cao, Xueru Bai, Feng Zhou, Deyu Meng, and Zhi Wang. SegEarth-OV: Towards training-free open-vocabulary segmentation for remote sensing images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10545–10556, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Li_SegEarth-OV_Towards_Training-Free_Open-Vocabulary_Segmentation_for_Remote_Sensing_Images_CVPR_2025_paper.html.

Xiangtai Li, Henghui Ding, Haobo Yuan, Wenwei Zhang, Jiangmiao Pang, Guangliang Cheng, Kai Chen, Ziwei Liu, and Chen Change Loy. Transformer-based visual segmentation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10138–10163, 2024. doi: 10.1109/TPAMI.2024.3434373.

Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-vit: Accurate and fully quantized low-bit vision transformer. *NeurIPS*, 35:34451–34463, 2022.

Ye Li, Chen Tang, Yuan Meng, Jiajun Fan, Zenghao Chai, Xinzhu Ma, Zhi Wang, and Wenwu Zhu. Prance: Joint token-optimization and structural channel-pruning for adaptive vit inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(1):283–298, 2026. doi: 10.1109/TPAMI.2025.3605239.

Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In *ICCV*, pp. 17227–17236, 2023.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*, 2022.

Yutong Lin, Yuhui Yuan, Zheng Zhang, Chen Li, Nanning Zheng, and Han Hu. DETR does not need multi-scale or locality design. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6545–6554, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Lin_DETR_Does_Not_Need_Multi-Scale_or_Locality_Design_ICCV_2023_paper.html.

Chenglong Lu, Shen Liang, Xuewei Wang, and Wei Wang. Reinforcement learning-based token pruning in vision transformers: A markov game approach. In *2025 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. IEEE, 2025.

Junzhu Mao, Yang Shen, Jinyang Guo, Yazhou Yao, Xiansheng Hua, and Hengtao Shen. Prune and merge: Efficient token compression for vision transformer with spatial information preserved. *TMM*, 2025.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, pp. 1928–1937. PmLR, 2016a.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, pp. 1928–1937. PmLR, 2016b.

Haolin Qin, Daquan Zhou, Tingfa Xu, Ziyang Bian, and Jianan Li. Factorization vision transformer: Modeling long-range dependency with local window cost. *TNNLS*, 2023.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems*, volume 34, pp. 13937–13949, 2021.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Weixuan Sun, Zhen Qin, Hui Deng, Jianyuan Wang, Yi Zhang, Kaihao Zhang, Nick Barnes, Stan Birchfield, Lingpeng Kong, and Yiran Zhong. Vicinity vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. doi: 10.1109/TPAMI.2023.3285569.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pp. 10347–10357. PMLR, 2021.

Hongjie Wang, Bhishma Dedhia, and Niraj K Jha. Zero-tprune: Zero-shot token pruning through leveraging of the attention graph in pre-trained transformers. In *CVPR*, pp. 16070–16079, 2024.

Zi Wang and Chengcheng Li. Channel pruning via lookahead search guided reinforcement learning. In *WACV*, pp. 2029–2040, 2022.

Ziwei Wang, Changyuan Wang, Xiuwei Xu, Jie Zhou, and Jiwen Lu. Quantformer: Learning extremely low-precision vision transformers. *TPAMI*, 45(7):8813–8826, 2022.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Siyuan Wei, Tianzhu Ye, Shen Zhang, Yao Tang, and Jiajun Liang. Joint token pruning and squeezing towards more aggressive compression of vision transformers. In *CVPR*, pp. 2092–2101, 2023.

Ross Wightman et al. Pytorch image models, 2019.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Jianzong Wu, Xiangtai Li, Xia Li, Henghui Ding, Yunhai Tong, and Dacheng Tao. Toward robust referring image segmentation. *IEEE Transactions on Image Processing*, 33:1782–1794, 2024. doi: 10.1109/TIP.2024.3371348.

Xuwei Xu, Sen Wang, Yudong Chen, Yanping Zheng, Zhewei Wei, and Jiajun Liu. GTP-ViT: Efficient vision transformers via graph-based token propagation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 86–95, 2024.

Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-ViT: Slow-fast token evolution for dynamic vision transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(3):2964–2972, 2022. doi: 10.1609/aaai.v36i3.20202.

Yiting Yang, Hao Luo, Yuan Sun, Qingsen Yan, Haokui Zhang, Wei Dong, Guoqing Wang, Peng Wang, Yang Yang, and Hengtao Shen. Efficient adaptation of pre-trained vision transformer underpinned by approximately orthogonal fine-tuning strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4878–4887, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Yang_Efficient_Adaptation_of_Pre-trained_Vision_Transformer_underpinned_by_Approximately_Orthogonal_ICCV_2025_paper.html.

Guangzhen Yao, Yuehan Wang, Hui Xu, Long Zhang, and MiaoQI MiaoQI. Global-pruner: A stable and efficient pruner for retraining-free pruning of encoder-based language models. In *Proceedings of the 28th Conference on Computational Natural Language Learning*, pp. 46–55, 2024.

Ting Yao, Yehao Li, Yingwei Pan, Yu Wang, Xiao-Ping Zhang, and Tao Mei. Dual vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10870–10882, 2023. URL https://ieeexplore.ieee.org/document/10105499/.

Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. TR-BERT: Dynamic token reduction for accelerating BERT inference. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5798–5809. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.463.

Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10809–10818, June 2022.

Lu Yu and Wei Xiang. X-Pruner: explainable pruning for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24355–24363, 2023.

Chang-Bin Zhang, Yujie Zhong, and Kai Han. Mr. DETR: Instructive multi-route training for detection transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Zhang_Mr._DETR_Instructive_Multi-Route_Training_for_Detection_Transformers_CVPR_2025_paper.html.

Qiming Zhang, Jing Zhang, Yufei Xu, and Dacheng Tao. Vision transformer with quadrangle attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3608–3624, 2024. doi: 10.1109/TPAMI.2023.3347693.

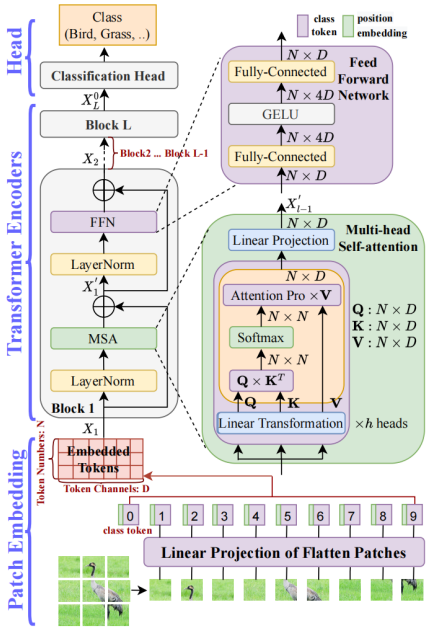


Figure 5: Basic architecture of a Vision Transformer.

A Preliminary and Background

A.1 Vision Transformer

A Vision Transformer (ViT) partitions an input image $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ into $N = HW/P^2$ non-overlapping patches of size $P \times P$. Each patch is projected by a linear layer into a D-dimensional Patch Token. A CLS Token is prepended to the Patch Token sequence, and positional embeddings are added to form the input sequence:

$$\mathbf{Z}_0 = \left[\mathbf{x}^0_{\text{cls}}; \mathbf{E}_0\right] + \mathbf{E}_{\text{pos}}, \qquad \mathbf{E}_0 \in \mathbb{R}^{N\times D}. \tag{12}$$

As illustrated in Figure 5, the input sequence is processed by a series of Transformer Blocks. Each Block models the relationships among Tokens through multi-head self-attention and then updates their representations using a feed-forward network. Finally, the representation of the CLS Token is passed to the classification head to produce the prediction.

A.2 Token Pruning

Traditional Token Pruning methods evaluate Patch Tokens at intermediate ViT layers according to attention responses, Token similarity, or other importance measures, and retain only a subset of the Tokens. The CLS Token is typically preserved throughout the network. Let the input Patch Token sequence at layer t be $\mathbf{E}_{t-1} \in \mathbb{R}^{N_{t-1}\times D}$, and let $\mathcal{K}_t$ denote the set of retained local indices. The retained Token sequence is written as

$$\widehat{\mathbf{E}}_{t-1} = \text{Gather}\left(\mathbf{E}_{t-1}, \mathcal{K}_t\right) \in \mathbb{R}^{N_t\times D}, \qquad N_t = |\mathcal{K}_t| \le N_{t-1}. \tag{13}$$

As illustrated in Figure 6, Token Pruning first determines the retained set according to a predefined importance rule and then processes the selected Patch Tokens in subsequent layers. Depending on the implementation, the selected Tokens can be physically gathered to form a shorter sequence, or masked and padded to preserve the original sequence length. These execution mechanisms lead to different reductions in the actual computation and inference latency of subsequent layers. The selected Token subset also affects the representations produced by later layers and the final prediction.

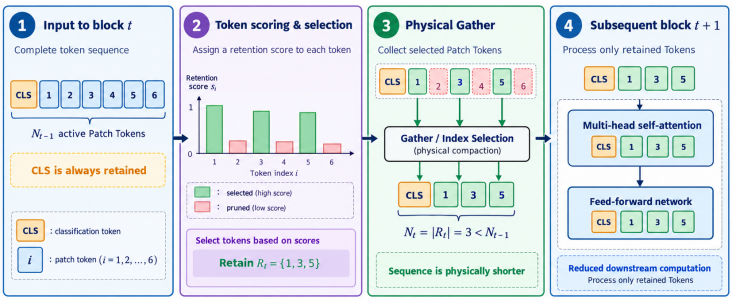


Figure 6: Illustration of the basic Token Pruning operation.

B Structural Correctness of FASTER

This appendix establishes the structural correctness of FASTER under the monotonically non-increasing Token-budget schedule. We prove that deterministic Top-k selection satisfies the prescribed number of Patch Tokens at every Transformer layer, preserves a nonempty and monotonically non-increasing Token sequence, and maintains the alignment between each surviving Token and its original image-patch identity after repeated physical Token compaction.

The proof concerns Patch Tokens only. The class Token is always retained and is excluded from the Patch-Token count and the Origin Index Map.

B.1 Definitions and Assumptions

Consider a frozen Vision Transformer with L Transformer blocks. Let N denote the initial number of Patch Tokens and let D be the feature dimension. Before the pruning decision at layer t, the surviving Patch Token features are represented by $E_{t-1} \in \mathbb{R}^{N_{t-1} \times D}$, where N_{t-1} is the number of surviving Patch Tokens.

The Origin Index Map is defined as $P_{t-1} \in \{1, \ldots, N\}^{N_{t-1}}$, $P_0 = [1, 2, \ldots, N]$. The entry $P_{t-1}[i]$ records the original image-patch identity associated with row i of E_{t-1}. Initially, $N_0 = N$.

FASTER uses a monotonically non-increasing layer-wise budget schedule: $1 \geq B_1 \geq B_2 \geq \cdots \geq B_L > 0$. The nominal number of Patch Tokens retained at layer t is $\bar{k}_t = \max(1, \lfloor B_t N \rfloor)$, $\bar{k}_0 = N$. The effective target count is $k_t = \min\left(\bar{k}_t, N_{t-1}\right)$.

Let $q_{t,i}$ denote the retention probability predicted by the factorized Bernoulli policy for the i-th surviving Token at layer t. During deterministic inference, FASTER selects the k_t Tokens with the largest retention probabilities:

$$\widetilde{\mathcal{K}}_t = \mathrm{TopKIdx}\left(\{q_{t,i}\}_{i=1}^{N_{t-1}}, k_t\right). \tag{14}$$

If two Tokens have the same retention probability, the normalized task-aware sensitivity score is used as the first tie-breaking criterion. Remaining ties are resolved according to the smaller original identity $P_{t-1}[i]$. Therefore, the selected subset is uniquely determined.

Before physical compaction, the selected local indices are sorted in ascending order:

$$\mathcal{K}_t = \mathrm{SortAsc}\left(\widetilde{\mathcal{K}}_t\right) = (i_1, \ldots, i_{k_t}), \qquad i_1 < \cdots < i_{k_t}. \tag{15}$$

FASTER applies the same ordered index set to the feature matrix and the Origin Index Map:

$$\widehat{E}_{t-1} = \mathrm{Gather}\left(E_{t-1}, \mathcal{K}_t\right), \qquad P_t = \mathrm{Gather}\left(P_{t-1}, \mathcal{K}_t\right). \tag{16}$$

The compacted Patch Tokens are concatenated with the class Token and processed by the frozen Transformer block:

$$[x_{\mathrm{cls}}^t, E_t] = \mathrm{Block}_t\left([x_{\mathrm{cls}}^{t-1}, \widehat{E}_{t-1}]\right). \tag{17}$$

A standard Transformer block updates the feature values but does not permute the sequence rows.

B.2 Monotonicity of the Nominal Token Budget

For every $t \in \{1, \dots, L\}$, $1 \le \bar{k}_t \le \bar{k}_{t-1} \le N$.

Proof. The lower bound follows directly from the definition:

$$\bar{k}_t = \max\left(1, \lfloor B_t N \rfloor\right) \ge 1. \tag{18}$$

For $t = 1$, since $B_1 \le 1$,

$$\bar{k}_1 = \max\left(1, \lfloor B_1 N \rfloor\right) \le N = \bar{k}_0. \tag{19}$$

For $t \ge 2$, the budget schedule gives $B_t \le B_{t-1}$. Since multiplication by the positive number N, the floor operation, and the maximum with 1 preserve monotonicity,

$$\bar{k}_t = \max\left(1, \lfloor B_t N \rfloor\right) \le \max\left(1, \lfloor B_{t-1} N \rfloor\right) = \bar{k}_{t-1}. \tag{20}$$

The upper bound $\bar{k}_{t-1} \le N$ follows recursively from $\bar{k}_0 = N$. □

B.3 Exact Token-Count Satisfaction

During deterministic FASTER inference, the number of surviving Patch Tokens after layer t satisfies

$$N_t = k_t = \bar{k}_t, \qquad t \in \{1, \dots, L\}. \tag{21}$$

Consequently, $1 \le N_t \le N_{t-1} \le N$.

Proof. We prove the result by induction.

Before the first pruning decision, $N_0 = N = \bar{k}_0$. By Lemma B.2, $\bar{k}_1 \le \bar{k}_0 = N_0$. Therefore,

$$k_1 = \min(\bar{k}_1, N_0) = \bar{k}_1. \tag{22}$$

The Top-k_1 operator selects exactly k_1 distinct local indices. Physical Gather consequently produces exactly k_1 Patch-Token rows. The Transformer block does not change the number of sequence rows, so

$$N_1 = k_1 = \bar{k}_1. \tag{23}$$

Now assume that $N_{t-1} = \bar{k}_{t-1}$ holds after layer $t-1$. By Lemma B.2, $\bar{k}_t \le \bar{k}_{t-1} = N_{t-1}$. Thus, $k_t = \min(\bar{k}_t, N_{t-1}) = \bar{k}_t$. Top-$k_t$ selects exactly k_t distinct indices, and the Gather operation constructs exactly k_t Patch-Token rows. Since the Transformer block preserves the number of rows, $N_t = |\mathcal{K}_t| = k_t = \bar{k}_t$. The lower and upper bounds follow from $\bar{k}_t \ge 1$ and $\bar{k}_t \le \bar{k}_{t-1} = N_{t-1}$. Therefore, the result holds for every layer. □

FASTER never constructs an empty Patch-Token sequence during deterministic inference. Including the class Token, every Transformer block processes at least two sequence rows.

Proof. Theorem B.3 gives $N_t \ge 1$ at every layer. The class Token is always retained separately. □

B.4 Order Preservation of the Origin Index Map

For every $t \in \{0, \dots, L\}$, the Origin Index Map is strictly increasing:

$$P_t[1] < P_t[2] < \cdots < P_t[N_t]. \tag{24}$$

Therefore, every surviving row has a unique original Patch identity and the original relative order is preserved.

Proof. At initialization, $P_0 = [1, 2, \dots, N]$, which is strictly increasing.

Assume that P_{t-1} is strictly increasing. Let

$$\mathcal{K}_t = (i_1, \dots, i_{N_t}), \qquad i_1 < \cdots < i_{N_t}. \tag{25}$$

By the definition of Gather,

$$P_t[j] = P_{t-1}[i_j], \qquad j = 1, \ldots, N_t. \tag{26}$$

Because P_{t-1} is strictly increasing and $i_1 < \cdots < i_{N_t}$,

$$P_{t-1}[i_1] < P_{t-1}[i_2] < \cdots < P_{t-1}[i_{N_t}]. \tag{27}$$

Hence,

$$P_t[1] < P_t[2] < \cdots < P_t[N_t]. \tag{28}$$

The property follows by induction. □

B.5 Identity Alignment after Physical Token Compaction

For every layer t and every surviving row $j \in \{1, \ldots, N_t\}$, the feature row $E_t[j]$ has a unique correspondence with its original Patch identity $P_t[j]$. Consequently, the original positional feature, task-aware sensitivity score, candidate indicator, and reference feature associated with that Patch can be retrieved using $P_t[j]$.

Proof. At initialization, row i of E_0 corresponds to the original Patch identity $P_0[i] = i$. Therefore, $E_0[i] \longleftrightarrow P_0[i]$ holds for every initial Patch Token.

Assume that before pruning layer t, $E_{t-1}[i] \longleftrightarrow P_{t-1}[i]$ holds for every surviving row i. Let $\mathcal{K}_t = (i_1, \ldots, i_{N_t})$. Because the same ordered indices are used for both feature compaction and identity compaction,

$$\widehat{E}_{t-1}[j] = E_{t-1}[i_j], \qquad P_t[j] = P_{t-1}[i_j]. \tag{29}$$

The induction hypothesis therefore gives $\widehat{E}_{t-1}[j] \longleftrightarrow P_t[j]$.

The Transformer block applies self-attention, residual connections, normalization, and the token-wise feed-forward network to the current sequence. These operations update feature values but do not permute sequence rows. Hence, the output row preserves the same identity anchor: $E_t[j] \longleftrightarrow P_t[j]$.

By Lemma B.4, $P_t[j]$ is a unique element of $\{1, \ldots, N\}$. It therefore uniquely identifies the corresponding original image-patch position $\mathrm{Pos}_{P_t[j]}$, the normalized task-aware sensitivity score $\widetilde{S}^{t+1}_{P_t[j]}$, the candidate indicator $\chi_{t+1,j} = \mathbb{I}\left(P_t[j]^{t+1}_{\mathrm{guide}}\right)$, and the corresponding reference feature $E_t^{\mathrm{base}}[P_t[j]]$. Thus, all identity-dependent lookups remain aligned with the correct surviving Token row. □

B.6 Validity during Stochastic Policy Training

The previous results concern deterministic Top-k inference. The identity alignment argument also applies during A3C training to any valid sampled action.

Let the factorized Bernoulli action at layer t be $a_t = (a_{t,1}, \ldots, a_{t,N_{t-1}}), a_{t,i} \in \{0, 1\}$.

A sampled action is valid if $\sum_{i=1}^{N_{t-1}} a_{t,i} > 0$.

For a valid action, define the selected local indices as

$$\mathcal{K}_t^{\mathrm{train}} = \mathrm{SortAsc}\left(\{i : a_{t,i} = 1\}\right). \tag{30}$$

The same ordered set is applied to E_{t-1} and P_{t-1}:

$$\widehat{E}_{t-1} = \mathrm{Gather}\left(E_{t-1}, \mathcal{K}_t^{\mathrm{train}}\right), \qquad P_t = \mathrm{Gather}\left(P_{t-1}, \mathcal{K}_t^{\mathrm{train}}\right). \tag{31}$$

Since the selected indices are distinct and sorted, the proof of Lemma B.4 applies without change. Therefore, for every valid sampled action, $E_t[j] \longleftrightarrow P_t[j]$ holds for every surviving row.

If an all-zero action is sampled, FASTER assigns the invalid-action penalty r_{invalid} and terminates the current episode before constructing an empty compact sequence. Consequently, the physical compaction operation and the reconstruction reward are evaluated only for valid nonempty Token sets.

For a valid action, the reconstruction reward can therefore be defined as

$$R_t^{\text{rec}} = -\frac{1}{N_t D} \left\| E_t - \text{Gather}\left(\text{sg}\left(E_t^{\text{base}}\right), P_t\right) \right\|_F^2, \tag{32}$$

where $\text{sg}(\cdot)$ denotes stop-gradient. Since $N_t \geq 1$ and $D > 0$, the normalization factor $N_t D$ is strictly positive, and the reward is well-defined.

B.7 Scope of the Guarantee

The above results establish the following structural properties of FASTER:

1. deterministic Top-k inference satisfies the prescribed layer-wise Patch-Token budget;
2. the number of surviving Patch Tokens is nonempty and monotonically non-increasing;
3. repeated Physical Gather operations preserve unique and order-consistent original Patch identities;
4. task-aware sensitivity, positional, candidate, and reference features are retrieved using the correct original identity; and
5. the reconstruction reward is well-defined for every valid stochastic training action.

These results are structural guarantees and do not establish that A3C converges to a globally optimal policy or that the task-aware sensitivity encoding identifies a globally optimal Token subset. They also do not imply that a theoretical GFLOPs reduction produces an identical proportional reduction in hardware latency. Such properties depend on policy optimization, empirical model performance, and the deployment environment.

C Theoretical Analysis of the Budget Consistency Reward

The budget consistency reward in FASTER encourages the stochastic Token selection policy to follow the prescribed layer-wise computational budget. At layer t, each surviving Token is independently retained according to a Bernoulli distribution:

$$a_{t,i} \sim \text{Bernoulli}(q_{t,i}), \qquad i = 1, \ldots, N_{t-1}, \tag{33}$$

where $q_{t,i}$ is the retention probability predicted by the Actor. The number of retained Tokens is therefore $N_t = \sum_{i=1}^{N_{t-1}} a_{t,i}$.

Let $\bar{k}_t$ denote the target number of retained Patch Tokens at layer t. We define the squared budget deviation as $D_t = (N_t - \bar{k}_t)^2$.

For a fixed state X_t, the conditional expectation of the squared budget deviation is

$$\mathbb{E}\left[(N_t - \bar{k}_t)^2 \mid X_t\right] = \left(\sum_{i=1}^{N_{t-1}} q_{t,i} - \bar{k}_t\right)^2 + \sum_{i=1}^{N_{t-1}} q_{t,i}(1 - q_{t,i}). \tag{34}$$

Proof. For a fixed state X_t, the quantities $q_{t,i}$ and $\bar{k}_t$ are constant. By the bias–variance decomposition,

$$\mathbb{E}\left[(N_t - \bar{k}_t)^2 \mid X_t\right] \left(\mathbb{E}[N_t \mid X_t] - \bar{k}_t\right)^2 + \text{Var}(N_t \mid X_t). \tag{35}$$

Because N_t is the sum of independent Bernoulli random variables,

$$\mathbb{E}[N_t \mid X_t] = \sum_{i=1}^{N_{t-1}} q_{t,i}, \qquad \text{Var}(N_t \mid X_t) = \sum_{i=1}^{N_{t-1}} q_{t,i}(1 - q_{t,i}).$$

Substituting these two expressions into the bias–variance decomposition gives

$$\mathbb{E}\left[(N_t - \bar{k}_t)^2 \mid X_t\right] = \left(\sum_{i=1}^{N_{t-1}} q_{t,i} - \bar{k}_t\right)^2 + \sum_{i=1}^{N_{t-1}} q_{t,i}(1 - q_{t,i}). \tag{36}$$

□

Table 7: Hardware and software environment used in the experiments.

Category	Component	Specification
Hardware	GPU	NVIDIA A100 80GB
	CPU	Intel Xeon Gold 6338
	Host memory	256GB
	Storage	2TB NVMe SSD
Software	Operating system	Ubuntu 20.04 LTS
	NVIDIA driver	535
	CUDA / cuDNN	11.8 / 8.6
	Python / PyTorch	3.9 / 2.0
	`timm`	0.9.2
	Inference precision	Automatic mixed precision

The first term, $\left(\sum_{i=1}^{N_{t-1}} q_{t,i} - \bar{k}_t\right)^2$, measures the deviation between the expected number of retained Tokens and the target budget. Minimizing this term encourages the Actor to allocate retention probabilities such that the expected computation remains close to the prescribed budget.

The second term, $\sum_{i=1}^{N_{t-1}} q_{t,i}(1 - q_{t,i})$, is the conditional variance of the sampled Token count. It quantifies the uncertainty introduced by stochastic Bernoulli exploration during policy training. Therefore, when the budget reward is defined as the negative squared budget deviation, $R_t^{\text{budget}} = -\left(N_t - \bar{k}_t\right)^2$, optimizing its expectation simultaneously encourages budget alignment and reduces fluctuations in the sampled Token count.

During deterministic inference, FASTER replaces Bernoulli sampling with Top-k_t decoding, where $k_t = \min(\bar{k}_t, N_{t-1})$. The inference procedure therefore retains exactly k_t Tokens. The budget consistency reward provides a soft training-time alignment between the expected stochastic behavior and the deterministic deployment rule, thereby reducing the mismatch between policy optimization and inference-time Token selection.

D Experimental Setup and Reproducibility Protocol

D.1 Hardware, Software, and Datasets

All experiments are conducted on the same workstation, which is equipped with an NVIDIA A100 80GB GPU, an Intel Xeon Gold 6338 CPU, 256GB of host memory, and a 2TB NVMe solid-state drive. FASTER is implemented in Python 3.9 and PyTorch 2.0. The experimental environment uses Ubuntu 20.04 LTS, NVIDIA Driver 535, CUDA 11.8, cuDNN 8.6, and timm v0.9.2 Wightman et al. (2019). All methods use the same hardware platform, software environment, numerical precision, and measurement protocol.

The primary experiments are conducted on ImageNet-1K Deng et al. (2009) for image classification. Downstream transfer experiments use seven classification datasets, including Flowers-102, Oxford-IIIT Pets, DTD, MIT Indoor-67, CUB-200, FGVC-Aircraft, and Stanford Cars, as well as two semantic segmentation datasets, Pascal VOC and COCO.

D.2 Data Partitions and Image Preprocessing

ImageNet-1K contains 1,000 classes, 1,281,167 training images, and 50,000 validation images. The training set is divided into three mutually exclusive subsets:

$$\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{cal}} \cup \mathcal{D}_{\text{policy}} \cup \mathcal{D}_{\text{select}}. \tag{37}$$

The calibration set $\mathcal{D}_{\text{cal}}$ contains 10,000 images, with 10 images sampled from each class, and is used only for TSE calibration. The selection set $\mathcal{D}_{\text{select}}$ contains 20,000 images, with 20 images sampled from each class, and is used for hyperparameter and policy-checkpoint selection. The remaining 1,251,167 training images are used for Actor–Critic policy optimization. The partition is generated by class-stratified sampling with random seed 2026 and remains fixed across different backbones, computational budgets, random seeds, and ablation experiments.

Table 8: ImageNet-1K data partitions and their purposes.

Data split	Images	Purpose
$\mathcal{D}_{\text{cal}}$	10,000	TSE calibration
$\mathcal{D}_{\text{policy}}$	1,251,167	Actor–Critic optimization
$\mathcal{D}_{\text{select}}$	20,000	Hyperparameter and checkpoint selection
ImageNet validation	50,000	Final evaluation only

Table 9: Backbone configurations and primary computational budgets.

Backbone	Blocks	Dim.	Heads	Patch	Params. (M)	Budget (G)
DeiT-T	12	192	3	16×16	5.7	0.6
DeiT-S	12	384	6	16×16	22.1	2.0
DeiT-B	12	768	12	16×16	86.6	15.0
ViT-L	24	1024	16	16×16	304.3	35.0

The official ImageNet validation set is used only for final evaluation. It is not used for TSE computation, policy optimization, reward-weight selection, hyperparameter search, or policy-checkpoint selection.

At the primary input resolution of 224×224, images used for TSE calibration and policy training are processed using random resized cropping with a crop-scale range of $[0.08, 1.0]$, followed by horizontal flipping with probability 0.5. For policy-checkpoint selection and final ImageNet evaluation, the shorter image side is first resized to 256 pixels, followed by a 224×224 center crop. Images are normalized using the mean and standard deviation associated with the corresponding pretrained `timm` checkpoint. Comparison methods using the same backbone adopt the same preprocessing, input resolution, numerical precision, and evaluation split.

D.3 Backbones, Budgets, and Trainable Modules

The primary experiments use four pretrained Vision Transformer backbones: DeiT-Tiny, DeiT-Small, DeiT-Base Touvron et al. (2021), and ViT-Large Dosovitskiy (2020). All models use 16×16 Patch Tokens and receive 224×224 input images.

FASTER freezes the Patch Embedding, CLS Token, positional embeddings, Transformer Blocks, normalization layers, and task head during both training and inference. During policy training, only the global feature projection, local Token feature projection, positional feature projection, Actor, Critic Token encoder, and Critic value head are updated. During inference, only the frozen ViT, fixed TSE, state-projection modules, Actor, Top-k, Physical Gather, Identity Mapping, and task head are retained.

All computational costs follow the convention that one multiplication and one addition count as two floating-point operations: $F_{\text{FLOP}} = 2F_{\text{MAC}}$.

D.4 TSE Calibration and Policy Architecture

TSE is computed on the frozen and uncompressed ViT. During calibration, the complete Token sequence is used, and virtual gates measure the sensitivity of each Patch Token representation to the task loss. Responses from all samples are aggregated over $\mathcal{D}_{\text{cal}}$ and normalized independently within each Transformer layer. The continuous sensitivity scores and the discrete guidance features derived from sensitivity ranking are used as auxiliary Actor observations rather than hard Token-retention constraints. TSE is computed once during calibration and remains fixed during policy training and inference.

The Actor state contains the continuous TSE score, discrete guidance feature, global and local Token representations, original Patch position, current layer depth, target budget, and current remaining-Token ratio. The global, local, and positional features are each projected to 64 dimensions, resulting in a state dimension of $d_{\text{state}} = 2 + 64 + 64 + 64 + 3 = 197$.

The Actor has the architecture $197 \to 256 \to 256 \to 1$.

GELU activations are used in the hidden layers. The Critic uses the shared state-projection modules to map Token features to 128-dimensional representations, aggregates them by mean pooling, and combines the result with the layer depth, budget state, and remaining-Token ratio for value estimation.

D.5 Token Decoding and A3C Training

FASTER adopts a monotonically non-increasing layer-wise budget schedule:

$$1 \geq B_1 \geq B_2 \geq \cdots \geq B_L > 0. \tag{38}$$

Here, B_t denotes the target retention ratio of the t-th Transformer Block with respect to the original number of Patch Tokens, N. The target retention number and the feasible retention number are defined as

$$\bar{k}_t = \max\left(1, \lfloor B_t N \rfloor\right), \qquad k_t = \min\left(\bar{k}_t, N_{t-1}\right). \tag{39}$$

During training, Token-retention actions are sampled from the Bernoulli probabilities produced by the Actor. During inference, deterministic Top-k_t decoding selects the k_t Tokens with the highest retention probabilities, followed by reordering according to their original sequence order. The CLS Token is always retained. If a sampled action removes all Patch Tokens, it is treated as an invalid action and receives a penalty.

FASTER uses asynchronous Advantage Actor–Critic (A3C) to optimize the policy. The layer-wise Token-selection process for each image forms one episode and uses an undiscounted return $\gamma = 1$.

Each Worker collects a rollout of length H. When an episode terminates or reaches the maximum rollout length, the Critic estimates the bootstrap value of the truncated trajectory, and multi-step returns and layer-level advantages are computed. The number of workers, rollout length, learning rates, entropy schedule, reward weights, and update budget are fixed across optimizer comparisons and are specified in the released configuration files. During policy training, only the Actor–Critic parameters and state-projection modules are updated, while the ViT, task head, and TSE remain frozen.

D.6 Reward Design and Policy Selection

FASTER uses representation-consistency, budget-consistency, and terminal task rewards to associate local Token decisions with long-term compression outcomes. The representation-consistency reward compares intermediate representations from the compressed path with the corresponding representations from the complete reference path. The budget-consistency reward constrains the number of retained Tokens during stochastic exploration:

$$R^t_{\text{bud}} = -w_b \left(\frac{N_t - \bar{k}_t}{N} \right)^2, \qquad w_b > 0. \tag{40}$$

The terminal task reward is computed from the difference between the task losses of the compressed and complete reference paths. If all Patch Tokens are removed at a layer, an invalid-action penalty is applied and the current episode is terminated. All rewards are treated as stop-gradient environmental signals for return estimation and Actor–Critic optimization.

Hyperparameters are selected only using $\mathcal{D}_{\text{select}}$, without using the official ImageNet validation set. The selected parameters include the learning rate, entropy regularization coefficient, Critic loss weight, reward weights, TSE injection coefficient, rollout length, and training length. The final policy checkpoint is selected according to Top-1 accuracy on the selection set. If multiple checkpoints achieve the same accuracy, the one with the higher mean episode return and fewer training updates is selected.

D.7 Evaluation, Statistical Reporting, and Ablation Protocol

All trainable methods are evaluated using five independent random seeds: $\mathcal{S} = \{0, 1, 2, 3, 4\}$.

The mean and sample standard deviation over the five runs are reported. Deterministic training-free methods and fixed uncompressed baselines are evaluated once and reported as point estimates; the stochastic Random selection baseline is repeated over the same five seeds. Policy adaptation time denotes Actor–Critic optimization and checkpoint selection, whereas TSE calibration is reported separately.

Inference is evaluated with batch size one. Each configuration first performs 200 warm-up iterations, followed by 1,000 timed iterations. Input tensors are preloaded onto the GPU, and device synchronization is performed before and after each timed measurement. End-to-end latency includes state construction, Actor inference, Top-k selection, Physical Gather, Identity Mapping updates, the compressed ViT forward pass, and the task head. Disk access, image decoding, and host-to-device data transfer are excluded.

Throughput is measured independently as the number of images processed per second with batch size one. The reported speedup is defined as

$$\text{Speedup} = \frac{\overline{T}_{\text{baseline}}}{\overline{T}_{\text{FASTER}}}. \tag{41}$$

The component ablations remove TSE, the reinforcement-learning policy, Identity Mapping, reward components, or Physical Gather, and also evaluate random Token selection. Execution-mechanism ablations compare Zero Masking, Padded Logical Removal, and Physical Gather using the same selected Token subsets. The optimizer comparison evaluates Random, Q-Learning, SARSA, REINFORCE, A2C, TRPO, PPO, and A3C. All settings other than the investigated component or optimizer remain unchanged.

D.8 Downstream Transfer and Visualization Protocol

Downstream transfer experiments use the ImageNet-trained DeiT-B, TSE, Actor, budget schedule, deterministic Top-k rule, Physical Gather, and Identity Mapping. These components remain frozen during downstream evaluation. TSE is not recomputed, the Actor–Critic policy is not retrained, and a new Token budget is not searched.

For downstream classification, only a new task-specific classification head is trained. For semantic segmentation on Pascal VOC and COCO, only the segmentation decoder and prediction layers are trained. Compressed Tokens are restored to their original spatial positions through Identity Mapping before being passed to the downstream decoder.

Token visualizations use the frozen DeiT-B and deterministic Top-k policy. Retained Tokens after the 3rd, 6th, and 9th Transformer Blocks are recorded and mapped back to the original image Patch grid using Identity Mapping. White regions indicate removed Patch Tokens. No smoothing, interpolation, region expansion, or manual post-processing is applied.

E Detailed Ablation Protocol

This appendix describes the shared stage-wise protocol and calibration procedure used by the ablation studies. The main component and execution ablations are reported in the paper, while finer-grained state, reward, and identity analyses are provided in Appendix G. Unless otherwise specified, all experiments use the same frozen ViT backbone, task head, computational budget, data partition, preprocessing pipeline, and evaluation protocol as the main experiments.

E.1 Stage-wise Training and Inference Protocol

FASTER contains four sequential stages. First, task-aware sensitivity encoding is computed once from the frozen and uncompressed ViT using the calibration subset $\mathcal{D}_{\text{cal}}$. Second, the Actor–Critic policy is optimized on $\mathcal{D}_{\text{policy}}$ with stochastic Bernoulli actions, while the ViT backbone, task head, and sensitivity encoding remain frozen. Third, the policy checkpoint is selected using $\mathcal{D}_{\text{select}}$. Finally, deterministic Top-k decoding and Physical Gather are used during inference and frozen-policy transfer.

Table 10: Stage-wise data usage, trainable modules, token-selection rules, and outputs in FASTER.

Setting	TSE Calibration	Policy Optimization	Deterministic Inference	Frozen-policy Transfer
Data source	$\mathcal{D}_{\text{cal}}$	$\mathcal{D}_{\text{policy}}$	ImageNet validation	Downstream datasets
Checkpoint selection	–	$\mathcal{D}_{\text{select}}$	–	Not used
ViT backbone	Frozen	Frozen	Frozen	Frozen
Task head	Frozen	Frozen	Frozen	Task-specific head or decoder
Trainable modules	None	State projections, Actor, Critic encoder and value head	None	Classification head or segmentation decoder
Training-only modules	Virtual gates	Critic, Bernoulli sampler, complete reference branch	–	–
Token action	No pruning	Bernoulli sampling	Deterministic Top-k_t	Deterministic Top-k_t
Physical Gather	No	Valid sampled actions	Yes	Yes
TSE status	Computed	Fixed	Fixed	Fixed
Primary output	**Layer–patch TSE matrix**	**Selected policy checkpoint**	**Accuracy and efficiency metrics**	**Downstream task metrics**

The Critic network, complete reference branch, virtual gates, and stochastic Bernoulli sampler are used only during training. During inference, only the frozen ViT, fixed sensitivity encoding, Actor network, budget-constrained Top-k decoder, Identity Mapping, and Physical Gather operation are retained. The complete stage-wise protocol is summarized in Table 10.

E.2 Task-aware Sensitivity Encoding

FASTER computes task-aware sensitivity on the complete token sequence of the frozen ViT. For each calibration image, a virtual scalar gate is introduced for every Patch Token only during gradient computation. The resulting first-order task responses are aggregated over $\mathcal{D}_{\text{cal}}$ and normalized independently within each Transformer layer.

The continuous sensitivity scores and the corresponding discrete guidance features are provided to the Actor as auxiliary observations. They are not used as hard token-retention constraints. After calibration, the sensitivity encoding remains fixed during policy optimization and inference.

For a token representation e_p^{t-1} at the original patch position p, the virtual gated representation is defined as

$$\widetilde{e}_p^{t-1} = g_p^{t-1} e_p^{t-1}. \tag{42}$$

The local task response is computed by

$$d_{m,p}^t = \left. \frac{\partial \mathcal{L}(x_m, y_m)}{\partial g_p^{t-1}} \right|_{g=1} = \left\langle \left. \frac{\partial \mathcal{L}(x_m, y_m)}{\partial \widetilde{e}_p^{t-1}} \right|_{g=1}, e_p^{t-1} \right\rangle. \tag{43}$$

The responses are aggregated over the calibration set to obtain the layer–patch sensitivity matrix S. The scores are normalized within each layer according to

$$\widetilde{S}_p^t = \frac{S_p^t - \mu_t}{\sigma_t + \epsilon}. \tag{44}$$

Given the guidance ratio β_{guide}^t, the top-ranked sensitive patch positions form the guidance set K_{guide}^t. For a surviving token with original identity $o_{t,i}$, the binary guidance feature is

$$\chi_{t,i} = \mathbb{I}\left(o_{t,i} \in K_{\text{guide}}^t\right). \tag{45}$$

The continuous sensitivity representation and binary guidance feature are used as auxiliary policy observations. They do not directly determine the retained token set.

F Selection of the Policy Optimizer

F.1 Why A3C Fits FASTER

FASTER formulates Token retention as a layer-wise sequential decision process. At each Transformer layer, the Actor predicts an independent Bernoulli retention probability for every surviving Token. This factorized policy represents a complete Token mask through Token-level decisions and

Table 11: Optimization methods compared for Token selection in FASTER.

Method	Token decision	Learned baseline	Trajectory updates	Consideration in FASTER
Random	Random keep/drop	No	None	No-learning reference.
Q-learning	Token-wise keep/drop values	Action values	Value-based	Approximates individual Token values rather than the value of a complete mask.
SARSA	Token-wise keep/drop values	Action values	On-policy value-based	Has the same token-wise approximation limitation.
REINFORCE	Bernoulli policy	No	Policy gradient	Supports stochastic masks without a learned value baseline.
A2C	Bernoulli policy	Yes	Synchronous	Uses a Critic baseline and synchronized updates.
TRPO	Bernoulli policy	Yes	Trust-region	Supports stochastic masks with a more involved update procedure.
PPO	Bernoulli policy	Yes	Clipped policy update	Provides a competitive policy-gradient baseline.
A3C	**Bernoulli policy**	**Yes**	**Asynchronous**	**Combines a Critic baseline, multi-step returns, and diverse trajectories.**

avoids constructing a categorical action space whose size grows exponentially with the number of Tokens.

The reward design imposes additional requirements on the optimizer. An early Token decision affects later-layer representations, computational budgets, and the final task objective. Therefore, the optimizer must propagate delayed feedback across multiple decision steps. A3C addresses this requirement with a Critic baseline and multi-step returns. The Critic reduces the variance of policy-gradient estimation, while multi-step returns transfer later-layer and terminal rewards to earlier decisions. In addition, asynchronous workers collect trajectories from different images and sampled Token masks, which improves the diversity of training experience.

The ViT backbone, task head, and TSE remain frozen during policy optimization. Consequently, the policy-learning environment is relatively stable, making asynchronous Actor–Critic optimization suitable for FASTER. These properties motivate the use of A3C under the proposed setting. They indicate structural compatibility with FASTER rather than universal superiority over all reinforcement-learning optimizers.

F.2 Alternative Optimization Methods

Table 11 summarizes the decision interface and main characteristics of the optimization methods considered in FASTER. Random selection is used as a no-learning reference. Q-learning and SARSA are implemented as token-wise neural value-function baselines, with separate values for keeping and dropping each Token. These approximations do not directly model the value of a complete layer-wise Token mask, whose final return depends on interactions among multiple Token decisions.

REINFORCE, A2C, TRPO, PPO, and A3C directly support the factorized Bernoulli policy. REINFORCE does not use a learned value baseline, A2C relies on synchronous updates, and TRPO and PPO introduce alternative policy-update mechanisms. A3C combines a Critic baseline, multi-step returns, and asynchronous trajectory collection, which matches the sequential and delayed-reward structure of FASTER.

F.3 Unified Optimizer Comparison Protocol

All learnable optimizers use the same frozen ViT backbone and task head, TSE features, Token-level keep/drop interface, reward function, layer-wise budget schedule, training data, checkpoint-selection rule, and evaluation protocol. Q-learning and SARSA output keep and drop values for each Token, whereas REINFORCE, A2C, TRPO, PPO, and A3C output factorized Bernoulli retention probabilities. Each trainable method is independently evaluated using five random seeds, and the mean and standard deviation are reported. Random selection is evaluated once as a deterministic no-learning reference.

F.4 Accuracy and Efficiency Comparison

Table 12 compares the accuracy, inference efficiency, and policy adaptation cost of the evaluated optimizers across three FASTER operating points. A3C achieves the highest Top-1 accuracy on all three backbones, reaching 71.85%, 79.42%, and 81.83% on DeiT-T, DeiT-S, and DeiT-B, respectively. Compared with PPO, A3C improves accuracy by 0.32, 0.28, and 0.27 percentage points.

The latency and throughput values are measured using the same batch size, Actor interface, budget-constrained decoding, and Physical Gather operation. The policy adaptation time reflects the ad-

Table 12: Comparison of reinforcement-learning optimizers across different backbones. Results are reported as mean ± standard deviation over five independent runs. Latency and throughput include Actor inference, Token selection, and Physical Gather.

Optimizer	DeiT-T, 0.6 GFLOPs				DeiT-S, 2.0 GFLOPs				DeiT-B, 15.0 GFLOPs			
	Top-1 (%)	Latency (ms)	Throughput (img/s)	Policy Time (GPU h)	Top-1 (%)	Latency (ms)	Throughput (img/s)	Policy Time (GPU h)	Top-1 (%)	Latency (ms)	Throughput (img/s)	Policy Time (GPU h)
Random	65.80 ± 0.18	1.40	714.3	–	73.40 ± 0.21	3.28	304.9	–	78.52 ± 0.16	8.16	122.5	–
Q-Learning	67.12 ± 0.15	1.48	675.7	2.8	74.06 ± 0.16	3.46	289.0	8.6	78.97 ± 0.12	8.52	117.4	28.5
SARSA	66.94 ± 0.17	1.47	680.3	2.4	73.88 ± 0.18	3.43	291.5	7.4	79.08 ± 0.14	8.47	118.1	24.0
REINFORCE	69.35 ± 0.12	1.45	689.7	2.1	76.21 ± 0.11	3.39	295.0	6.2	80.32 ± 0.09	8.39	119.2	19.5
A2C	70.91 ± 0.08	1.43	699.3	1.1	78.62 ± 0.07	3.35	298.5	3.8	81.08 ± 0.06	8.31	120.3	9.8
TRPO	71.10 ± 0.07	1.43	699.3	1.0	78.90 ± 0.08	3.34	299.4	3.2	81.30 ± 0.05	8.28	120.8	7.1
PPO	71.53 ± 0.05	1.42	704.2	0.8	79.14 ± 0.05	3.32	301.2	2.5	81.56 ± 0.04	8.24	121.4	5.4
A3C (FASTER)	$\mathbf{71.85 \pm 0.03}$	**1.41**	**720.2**	**0.1**	$\mathbf{79.42 \pm 0.03}$	**3.30**	**305.0**	**0.3**	$\mathbf{81.83 \pm 0.02}$	**8.20**	**122.5**	**1.0**

Table 13: Ablation of policy-state components on DeiT-B at 15 GFLOPs.

State Configuration	Top-1 Accuracy (%)	Change (p.p.)
Without TSE Features	81.24 ± 0.05	-0.59
Without Global Context	81.14 ± 0.04	-0.69
Without Local Token Feature	80.71 ± 0.05	-1.12
Without Original Positional Feature	81.26 ± 0.04	-0.57
Without Layer-depth Cue	81.57 ± 0.03	-0.26
Without Budget Cue	81.05 ± 0.05	-0.78
Without Remaining-token Cue	81.45 ± 0.04	-0.38
Without Depth and Budget Cues	80.61 ± 0.05	-1.22
Full State	$\mathbf{81.83 \pm 0.02}$	–

ditional optimization required by each trainable method, while Random does not require policy training. A3C provides the most favorable combination of accuracy, inference efficiency, and policy adaptation cost under the tested configuration.

G Additional Ablation Studies

All additional ablations use the same frozen backbone, task head, TSE calibration, data partitions, budget schedule, numerical precision, and evaluation protocol as the main experiments. Unless stated otherwise, only the listed component is changed. Trainable variants use five seeds, while latency measurements use batch size one and include Actor inference, Token selection, Identity Mapping, Physical Gather, the compressed ViT, and the task head.

G.1 Policy-state Ablation

FASTER constructs the Actor observation from TSE features, global and local Token representations, original Patch positions, layer depth, budget state, and the remaining Token ratio. To evaluate the contribution of these components, we remove one state feature at a time while keeping the Actor architecture and training protocol unchanged.

The full policy state achieves the highest accuracy. Removing local Token features or both depth- and budget-related cues causes the largest degradation, showing that FASTER benefits from jointly modeling Token content, spatial identity, and the current compression state. The "Without TSE Features" variant removes only TSE inputs from the policy state; it is therefore distinct from the full-module TSE ablation reported in Table 3.

G.2 Cross-layer Long-term Reward

FASTER uses intermediate representation and budget feedback together with a terminal task reward. Table 14 evaluates the role of long-term reward propagation by comparing the complete reward design with immediate-only feedback and a variant without cross-layer long-term rewards.

Table 14: Ablation of cross-layer long-term rewards in FASTER.

Variant	GFLOPs	Top-1 (%)	Budget Error (%)	Policy Time (h)	Throughput (img/s)
		DeiT-T			
Immediate Reward Only	0.6	70.88 ± 0.08	4.6	0.2	682.5
Without Long-term Reward	0.6	71.14 ± 0.06	3.2	0.2	697.8
FASTER	0.6	**71.85 ± 0.03**	**1.8**	**0.1**	**720.2**
		DeiT-S			
Immediate Reward Only	2.0	78.69 ± 0.09	4.9	0.4	292.4
Without Long-term Reward	2.0	78.97 ± 0.07	3.5	0.4	298.1
FASTER	2.0	**79.42 ± 0.03**	**2.0**	**0.3**	**305.0**
		DeiT-B			
Immediate Reward Only	15.0	81.18 ± 0.08	5.2	1.2	119.0
Without Long-term Reward	15.0	81.46 ± 0.06	3.8	1.1	120.7
FASTER	15.0	**81.83 ± 0.02**	**2.1**	**1.0**	**122.5**

Table 15: Ablation of reward components on DeiT-B at 15 GFLOPs.

Variant	Top-1 (%)	Token Error (%)	Invalid Rate (%)	Latency (ms)	Policy Time (h)
Without Reconstruction Reward	80.57 ± 0.06	1.9	0.7	8.27	1.1
Without Budget Reward	80.99 ± 0.05	8.7	3.9	8.31	1.2
Without Terminal Task Reward	81.25 ± 0.05	2.0	0.8	8.25	1.1
Full Reward Design	**81.83 ± 0.02**	**1.8**	**0.6**	**8.20**	**1.0**

Using only immediate rewards leads to the largest budget deviations and the lowest accuracy. Removing long-term reward propagation partially improves budget alignment but still decreases Top-1 accuracy. The complete reward design allows early Token decisions to account for later representations and the final task objective.

G.3 Reward-component Ablation

We further evaluate the three reward components separately. The representation reward provides intermediate feature-preservation feedback, the budget reward constrains the stochastic Token count, and the terminal task reward connects the complete Token trajectory to classification performance.

Removing the budget reward produces the largest Token-count error and invalid-action rate, confirming its role in stabilizing budget-constrained exploration. Removing the terminal task reward mainly affects classification accuracy, while the reconstruction reward provides complementary intermediate representation feedback.

G.4 Identity Mapping Ablation

Identity Mapping records the original Patch identity of each surviving Token. This information is required for sensitivity lookup, positional feature extraction, and reference representation alignment after repeated physical compression.

The complete Identity Mapping consistently achieves the best accuracy. These results show that preserving original Patch identities is necessary for maintaining aligned task guidance and reference representations after successive Token compression.

G.5 Summary of Additional Ablations

The additional ablations support four conclusions. First, TSE and the complete policy state provide complementary task and instance information. Second, cross-layer reward propagation improves the assignment of delayed feedback to early Token decisions. Third, the three reward components control different aspects of policy behavior, including representation preservation, budget stability,

Table 16: Ablation of Identity Mapping across different backbone scales.

Variant	DeiT-T 0.6G	DeiT-S 2.0G	DeiT-B 15.0G	ViT-L 35.0G
Without Identity Mapping	70.42 ± 0.06	78.45 ± 0.06	81.50 ± 0.03	89.10 ± 0.09
Local-index Substitution	70.68 ± 0.06	78.66 ± 0.05	81.61 ± 0.03	89.28 ± 0.08
Misaligned TSE Query	70.71 ± 0.06	78.52 ± 0.06	81.35 ± 0.04	89.20 ± 0.08
Misaligned Reference Query	70.18 ± 0.07	78.06 ± 0.07	81.18 ± 0.04	89.05 ± 0.09
Full Identity Mapping	$\mathbf{71.85 \pm 0.03}$	$\mathbf{79.42 \pm 0.03}$	$\mathbf{81.83 \pm 0.02}$	$\mathbf{89.95 \pm 0.07}$

and final task performance. Finally, Identity Mapping is required to maintain correct spatial and reference-feature alignment after physical Token compression.