Learning Hierarchical Patch Splitting Policies for Faster Vision Transformers

Aditya Gupta[1] **Jean-Sebastien Dandurand**[1] **Kai Qiu**[1] **Rohan Choudhury**[1]
László A. Jeni[1]
[1]Carnegie Mellon University

Abstract

Vision Transformers (ViTs) are bottlenecked by the length of their input token sequence. Standard patchification fixes this length with a single patch size applied across the image, regardless of regional content. Prior work on adaptive patch sizing reduces token count by assigning coarser patches to visually simple regions, but relies on noisy pixel-level heuristics such as local entropy to make patchification decisions. We argue that this compute allocation should be grounded in the model's interpretation of task relevant information, not in raw pixel statistics. We introduce **S**plit **P**olicy **L**earned **I**mage **T**okenization which learns to allocate token resolution based on what the model finds meaningful, producing shorter token sequences that are better matched to task-relevant content. **SPLIT** matches the accuracy of full-resolution ViT classification baselines while using 30% fewer tokens. The same gains hold on dense prediction: **SPLIT** achieves competitive accuracy on COCO and ADE20K, demonstrating that semantically aligned patchification generalises well beyond classification.

1 Introduction

Vision Transformers (ViTs) [Dosovitskiy et al., 2021] have become the dominant architecture for various visual tasks, demonstrating strong scalability and effectiveness across a wide range of applications. However, their efficiency remains a major challenge: self-attention scales quadratically with sequence length, and because patch embedding uses a uniform grid, sequence length itself grows quadratically with input resolution. As a result, doubling the resolution of an image quadruples its token count and increases self-attention cost by roughly $16\times$, making high-resolution inputs prohibitively expensive to process.

To reduce the number of tokens processed while preserving task-relevant information, many prior works reduce the token count after the tokens have been fully computed. These methods typically leverage signals produced by the Vision Transformer itself, such as self-attention scores [Liang et al., 2022] or intermediate feature representations [Bolya et al., 2023], to filter, merge, or refine tokens [Rao et al., 2021], aligning token allocation with the model's learned semantics.

While promising, these methods are limited in their efficiency gains due to the cost of generating and processing dense tokens in the early stages. An alternative approach is to perform adaptive tokenization at the patch embedding stage itself. For example, Quadformer [Ronen et al., 2023] uses the feature discrepancy between an image and its downsampled-then-upsampled counterpart as a proxy for regional importance, and APT [Choudhury et al., 2026] uses entropy as a heuristic to estimate task-relevant uncertainty, allocating higher resolution to regions that are more informative for dense visual tasks.

Although adaptive patchification has proven effective, existing methods still rely on hand-crafted or external proxies to estimate regional importance. Visual signals such as entropy provide a useful

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



Original Images | Split Policy Learned Image Tokenizer (Ours)

Figure 1: **S**plit **P**olicy **L**earned **I**mage **T**okenization. SPLIT is an adaptive patchification framework that learns to allocate fine tokens to semantically important regions and coarse tokens to homogeneous ones, accelerating ViT inference without sacrificing accuracy.

estimate of local uncertainty or information density, but they are not directly optimized for the downstream task and may overlook semantically important regions while mistakenly assigning high importance to visually complex but task-irrelevant areas. The natural alternative, using semantic importance derived from the ViT itself, is impractical, since it requires first processing the full dense token sequence to obtain reliable importance signals. This leads to a key question: *can we obtain a lightweight yet semantic importance signal before processing with ViT layers?*

Our key insight is that this dilemma is less rigid than it appears. Although processing the full dense token sequence is expensive, coarse tokens provide a lightweight alternative for obtaining model-aligned semantic cues. By optimizing the patch selection policy through the downstream task objective, the errors in the final prediction relative to different patch selection policies implicitly reveal which regions are semantically important, allowing the splitting decisions themselves to be learned in a task-aware manner, without requiring external proxies or costly dense ViT guidance.

Building on this insight, we propose **S**plit **P**olicy **L**earned **I**mage **T**okenization (SPLIT), an adaptive patchification framework that learns task-aware patch selection from coarse to fine. As shown in fig. 1, SPLIT uses coarse tokens as lightweight semantic anchors and selectively refines only the regions that are semantically important to the downstream task. To further encourage efficiency, we formulate patch refinement as a budget-constrained decision-making problem and optimize the network with reinforcement learning, learning effective region selection without exceeding the prescribed token budget.

Our contributions can be summarized as follows:

- We introduce **SPLIT**, an adaptive patchification framework that learns task-aware patch selection from coarse semantic anchors, rather than pixel-level heuristics.
- We formulate adaptive patch refinement as a budget-constrained decision-making problem and optimize the decision policy with reinforcement learning, enabling effective region selection under a fixed token budget.
- SPLIT is lightweight to train: it adapts pretrained ViTs to semantically aligned tokenization with minimal fine-tuning, and transfers cleanly across image classification, object detection, semantic segmentation, and VQA benchmarks.

2 Related Works

2.1 Efficient Vision Transformer

Vision Transformers (ViTs) [Dosovitskiy et al., 2021, Touvron et al., 2021, Liu et al., 2021] have become a standard architecture for visual backbones across diverse domains, including image classification [Dosovitskiy et al., 2021], object detection [Carion et al., 2020], semantic segmentation [Strudel et al., 2021], and vision-language tasks [Radford et al., 2021]. Despite their strong performance and scalability, ViTs typically rely on a fixed patchification protocol: an input image is uniformly

divided into non-overlapping patches, each of which is linearly projected into a visual token. This protocol makes the input sequence length depend primarily on image resolution rather than image content, assigning the same token budget to redundant background regions, repetitive textures, and semantically important objects.

2.2 ViT Tokens Reduction

Due to the quadratic complexity of self-attention with respect to sequence length [Vaswani et al., 2017], reducing the number of visual tokens has become an important direction for improving ViT efficiency. Existing methods can be roughly divided into two categories. The first category reduces tokens after patchification, using intermediate model signals such as attention scores [Liang et al., 2022, Yin et al., 2022], token similarity [Bolya et al., 2023], or auxiliary importance predictors [Rao et al., 2021, Meng et al., 2022, Pan et al., 2021] as proxies for token relevance. These methods prune less informative tokens [Rao et al., 2021, Liang et al., 2022, Yin et al., 2022, Kong et al., 2022, Tang et al., 2022] or merge redundant token representations [Bolya et al., 2023, Lee and Hong, 2024, Chen et al., 2023b, Wu et al., 2023], allowing later transformer layers to operate on shorter sequences. While effective, these methods reduce tokens only after dense processing and introduce dynamic indexing or gather-scatter operations that can limit practical gains from optimized attention kernels such as FlashAttention [Dao et al., 2022]. The other category reduces tokens before transformer processing by adapting the patchification process itself. APT [Choudhury et al., 2026] and related adaptive patch sizing methods [Havtorn et al., 2023, Ronen et al., 2023, Chen et al., 2023a] assign larger patches to visually simple regions and smaller patches to visually complex regions, producing shorter input token sequences without mid-layer token manipulation. However, these methods typically rely on hand-crafted pixel-level heuristics and task-specific threshold tuning, which increases adaptation cost for each downstream setting.

2.3 Semantic Token Patchification

Besides requiring task-specific thresholds for patchification, current adaptive patch sizing methods rely primarily on low-level pixel statistics to determine patch granularity. For example, local entropy or visual complexity is often used as a proxy for where fine-grained tokens should be allocated. However, such signals do not necessarily reflect semantic importance [Hwang et al., 2025]. As illustrated in fig. 7, APT-style methods can over-tokenize noisy backgrounds or repetitive textures, while smooth but semantically meaningful regions can be compressed too aggressively. To address this limitation, our method learns a semantic patchification policy from the model's own feature representations.

3 Method

SPLIT is a learnable adaptive patchification framework for efficient ViTs. Given an image $I \in \mathbb{R}^{3\times H\times W}$, where H and W denote the image height and width, a vanilla ViT uniformly partitions it into $N = HW/p^2$ fixed-size patches, where p is the patch size and N is the resulting number of visual tokens. In contrast, SPLIT learns to generate only approximately ρN tokens under a token budget ratio $\rho \in (0, 1]$.

3.1 Problem Definition

Following prior work [Ronen et al., 2023, Choudhury et al., 2026], we use a multi-scale patch embedding function that maps patches of different spatial sizes into a shared token embedding space, allowing adaptive patches of varying resolutions to be processed by the same ViT backbone.

For a tokenizer with S stages and smallest patch size p, we first divide the image into patches of size $2^S p \times 2^S p$. A decision network then determines, based on each patch's content, whether to split it into four sub-patches of size $2^{S-1}p \times 2^{S-1}p$, and this process is applied recursively to every split patch over S rounds. The result is a mixed-resolution tokenization that uses larger patches for homogeneous regions and smaller patches for content-rich ones.

The decision network must trade off two conflicting objectives: maximizing downstream performance (which favors more, finer tokens) and minimizing compute per sample (which favors fewer, coarser tokens). We formulate this as a constrained optimization problem and train the decision network with reinforcement learning, minimizing the unconstrained Lagrangian dual of the constraint.

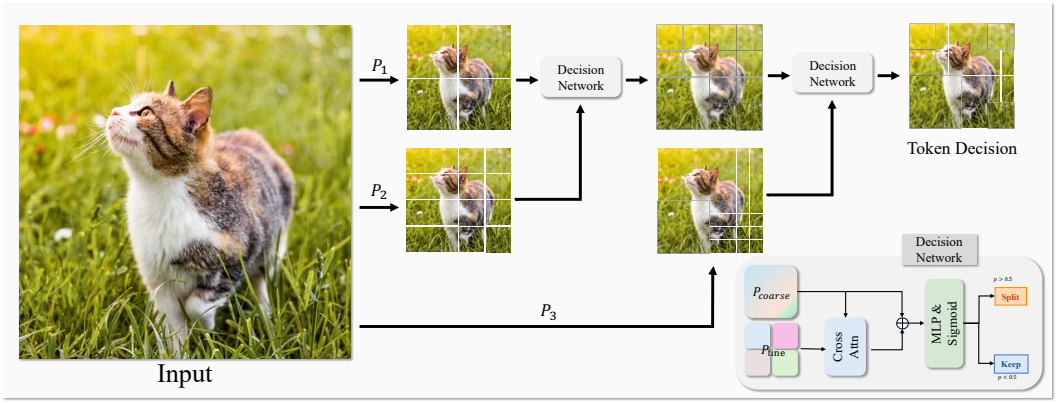


Figure 2: **Overview of SPLIT.** An input image is first divided into coarse patches. A decision network then performs S refinement stages, deciding whether to split each parent patch into its children or keep it as a single token. The example uses two splitting stages, yielding a mixed-resolution token sequence. For each parent-child group, a lightweight cross-attention module pools the four child embeddings with the parent embedding as the query. The pooled summary is concatenated with the parent embedding and passed to a two-layer MLP to predict the split probability p_i.

3.2 Decision Network

Consider a parent patch $P_i \in \mathbb{R}^{p\times p}$ with pre-positional embedding $z_i^p \in \mathbb{R}^{d_{embed}}$, and let $z_{i,1}^c, \ldots, z_{i,4}^c \in \mathbb{R}^{d_{embed}}$ be the embeddings of its four child sub-patches. All embeddings are computed using the multi-scale patch aggregation scheme of APT [Choudhury et al., 2026].

The decision network is a lightweight cross-attention MLP that uses the parent as a query and its children as keys and values, predicting the probability of splitting the parent into its children:

$$p_i = \sigma\big(h_\theta\big([\, z_i^p;\ \mathrm{CrossAttn}(z_i^p, \{z_{i,j}^c\}_{j=1}^4)\,]\big)\big)\,, \tag{1}$$

where h_θ is a two-layer MLP head and σ is the sigmoid. The attention pools the four child embeddings into a single summary, and the head decides whether that summary differs sufficiently from the parent to justify the finer tokens.

During training we sample the binary splitting decision $s_i \sim \mathrm{Bernoulli}(p_i)$, where $s_i = 0$ keeps the parent and $s_i = 1$ replaces it with its four children. At inference we use the deterministic rule $s_i = \mathbf{1}[p_i > 0.5]$. For a single splitting stage this yields the effective token set

$$\mathcal{T}(x, s) = \{z_i^p : s_i = 0\} \cup \{z_{i,j}^c : s_i = 1,\ j = 1, \ldots, 4\}. \tag{2}$$

With two stages, patches are organized into three scales (top, mid, fine), and the same network is applied at both transitions to decide whether to split each parent into its children. The formulation extends naturally to any number of stages.

3.3 PPO Training with Lagrange Dual

Training the decision network with ordinary supervised learning is not straightforward. In SPLIT, each coarse image patch must decide whether it should remain a single token or be split into finer subpatch tokens. This indexing operation is non-differentiable, so the decision network cannot be directly optimized through the downstream ViT loss by standard backpropagation. This non-differentiability can be addressed through differentiable relaxations, MSViT [Havtorn et al., 2023] makes use of Straight-Through Estimators and Gumbel-Sigmoid, however such relaxations either need to materialize all coarse and fine tokens at training time or need to prune un-materialized tokens which results in the un-selected path not getting any gradients. Due to these limitations we approach this problem using policy gradients and avoid using biased differentiable relaxations. We provide more detail on this decision in Appendix section B. Following the RL formulation used in adaptive routing methods such as Ada-K [Yue et al., 2025], we formulate patch splitting as a one-step reinforcement learning problem.

Consider a single stage of splitting, let $x \in \mathcal{X}$ be an input image and let $\mathcal{P}(x) = \{p_i\}_{i=1}^N$ denote the set of coarse patches. The decision network π_θ outputs a Bernoulli splitting distribution for each

coarse patch, $\pi_\theta(m_i = 1 \mid p_i), m_i \in \{0, 1\}$,, where $m_i = 1$ means that patch p_i is split into its child subpatches, and $m_i = 0$ means that p_i is retained as a coarse token. The full splitting mask is $M = (m_1, \ldots, m_N) \sim \pi_\theta(\cdot \mid x)$. A sequence packing operation $\mathcal{T}(x, M)$ uses the mask to create the final token sequence combining the fine tokens with the un-split coarse tokens retained by M. $\mathcal{T}(x, M)$ is now the input to the vision transformer backbone f_ϕ.

For a sampled mask M, we compute the downstream error signal $\mathcal{E}(x, M)$ and the fraction of sequence length relative to using all fine patches, $\mathcal{P}(x, M) = |\mathcal{T}(x, M)|/|\mathcal{T}_{\text{fine}}(x)|$, where $\mathcal{T}_{\text{fine}}(x)$ denotes the fully split patchify. The decision network has two conflicting objectives: it should preserve downstream task performance while using as few tokens as possible. We write the constrained objective as

$$\min_\theta \; J_{\text{err}}(\theta) \quad \text{s.t.} \quad J_{\text{tok}}(\theta) \le \rho, \tag{3}$$

where $J_{\text{err}}(\theta) = \mathbb{E}_{x,M}[\mathcal{E}(x, M)]$ and $J_{\text{tok}}(\theta) = \mathbb{E}_{x,M}[\mathcal{P}(x, M)]$ are taken over $x \sim \mathcal{D}$, $M \sim \pi_\theta(\cdot \mid x)$, and ρ is the target sequence length fraction. This is analogous to the reward-cost formulation in Safe RLHF [Dai et al., 2024], where the main objective is optimized subject to an expected cost constraint.

Using a Lagrangian relaxation, we optimize

$$\mathcal{L}(\theta, \lambda) = J_{\text{err}}(\theta) + \lambda\big(J_{\text{tok}}(\theta) - \rho\big), \qquad \lambda \ge 0.$$

For policy optimization, the constant term $-\lambda\rho$ does not affect gradients, we also normalize the combined cost by $(1 + \lambda)^{-1}$ to keep the scale of the cost stable as λ changes.

$$C(x, M; \lambda) = \frac{\mathcal{E}(x, M) + \lambda\mathcal{P}(x, M)}{1 + \lambda}.$$

For each image x, we sample L masks $M^{(1)}, \ldots, M^{(L)} \sim \pi_{\theta_{\text{old}}}(\cdot \mid x)$. We use a leave-one-out advantage comparing each of the ℓ rollouts to the other rollouts from the same image:

$$\widehat{A}_\ell = -(C_\ell - \frac{1}{L-1}\sum_{j \ne \ell} C_j) \qquad \text{where,} \quad C_\ell = \frac{\mathcal{E}(x, M^{(\ell)}) + \lambda\mathcal{P}(x, M^{(\ell)})}{1 + \lambda}.$$

Thus, a rollout has a positive advantage if its combined cost is less than that of the other sampled masks. We then optimize the decision network with a PPO clipped objective. Let $r_\ell(\theta) = \frac{\pi_\theta(M^{(\ell)}|x)}{\pi_{\theta_{\text{old}}}(M^{(\ell)}|x)}$ be the importance ratio, the PPO loss is

$$\mathcal{L}_{\text{PPO}}(\theta) = -\mathbb{E}_{x,\ell}\left[\min\left(r_\ell(\theta)\widehat{A}_\ell, \text{clip}\left(r_\ell(\theta), 1 - \epsilon, 1 + \epsilon\right)\widehat{A}_\ell\right)\right].$$

This objective allows the decision network to learn from discrete masks while preventing very large policy updates. Note that $\mathcal{L}_{\text{PPO}}$ does not explicitly guide the decision network towards ρ, instead the multiplier λ decides the relative importance of the two conflicting objectives and this decides the optimal $\mathbb{E}_{x\sim\mathcal{D},\, M\sim\pi_\theta(\cdot|x)}\big[\mathcal{P}(x, M))\big]$ corresponding to minimizing $\mathbb{E}_{x\sim\mathcal{D},\, M\sim\pi_\theta(\cdot|x)}\big[C(x, M; \lambda)\big]$

Effectively λ is the only source of control on how aggressively our decision network chooses to split patches, if λ is too small, the decision network may split nearly all patches and ignore the token budget. If λ is too large, the network may retain too few fine tokens and collapse to an overly cheap representation. Therefore, following the dual descent update used in constrained RL [Dai et al., 2024], we update λ independently of the PPO step to ensure that λ pushes the decision network to respect our budget ρ:

$$\lambda_{t+1} = \Big[\lambda_t + \eta_\lambda\big(\overline{P}_t - \rho\big)\Big]_+, \quad \text{where} \quad \overline{P}_t = \frac{1}{BL}\sum_{b=1}^{B}\sum_{\ell=1}^{L}\mathcal{P}(x_b, M_b^{(\ell)}),$$

B is the batch size, η_λ is the dual learning rate, and $[\cdot]_+ = \max(\cdot, 0)$ enforces non-negativity.

3.4 Decision Network Warm-start

Arbitrary initialization of the decision network can make RL training unstable. To reduce this instability, we warm-start the decision network with deterministic pseudo-labels derived from patch-subpatch similarity. Let z_i^c be the embedding of a coarse patch p_i and $\{z_{i,k}^f\}_{k=1}^4$ be the embeddings of its four child subpatches. For each parent–child pair, we compute the cosine dissimilarity $d_{i,k} =$

$(1 - \cos(z_i^c, z_{i,k}^f))/2 \in [0, 1]$. The pseudo-label $\widetilde{m}_i = \mathbf{1}[\max_k d_{i,k} > \tau_{\text{split}}]$ marks a parent for splitting if at least one of its children is sufficiently dissimilar from it, where τ_{split} is a learnable threshold. The decision network is trained to imitate these pseudo-labels with a binary cross-entropy loss, and the threshold τ_{split} is updated by dual ascent on the same budget gap $\overline{P}_t - \rho$ used during PPO training, so that the warm-start labels themselves track the target token budget.

During warm-start, these cosine-similarity masks are also used to fine-tune the ViT backbone, allowing the backbone to adapt to the adaptive token layout before the policy is optimized with PPO. In all subsequent decision network training, the ViT backbone is frozen.

4 Experiments

4.1 Image Classification

We provide results from training [1] with SPLIT for 1 epoch of warm-start followed by 1 epoch of PPO training initialized from a checkpoint already finetuned on ImageNet [Deng et al., 2009]. We provide a comparison against baselines including token merging methods like EViT, ToMe and DTEM on ViT-L/14 at a resolution of 224 in fig. 3. For a fair comparison in throughput, all the baselines in fig. 3 use flash attention. In tables 1 and 2, we show that our method is capable of matching performance to APT and vanilla ViT with significant speedup at an image size of 224 and 336.

Model	Res/Patch	Img/s	Speedup	Acc
ViT-B	224/16	3310	-	85.1
Random	224/16	3751	+13%	83.7
APT-B	224/16	3540	+7%	85.1
SPLIT-B (Ours)	224/16	3635	+10%	85.0
ViT-L	224/14	883	-	87.9
Random	224/14	1049	+19%	86.9
APT-L	224/14	993	+12%	87.8
SPLIT-L (Ours)	224/14	1107	+25%	87.8
ViT-H	224/14	441	-	88.3
Random	224/14	568	+29%	87.4
APT-H	224/14	542	+23%	88.3
SPLIT-H (Ours)	224/14	533	+21%	88.2

Table 1: **ImageNet Classification at 224.** Comparison of ViT models with different patch reduction strategies at 224x224 resolution.

Model	Res/Patch	Img/s	Speedup	Acc
ViT-L	336/14	395	-	88.2
Random	336/14	550	+39%	87.3
APT-L	336/14	527	+33%	88.1
SPLIT-L (Ours)	336/14	570	+44%	88.1
ViT-H	336/14	175	-	88.5
Random	336/14	272	+55%	87.0
APT-H	336/14	263	+50%	88.4
SPLIT-H (Ours)	336/14	281	+61%	88.4

Table 2: **ImageNet Classification at 336.** Comparison of ViT models with different patch reduction strategies at 336x336 resolution.

4.2 Dense Visual Tasks

Object detection. Object detection is a strong stress test for adaptive patchification because localization depends on preserving fine spatial detail at high resolution. To evaluate this setting, we train EVA-02 backbones [Fang et al., 2024] with a ViTDet-style detection head [Li et al., 2022] under window attention, and benchmark on COCO [Lin et al., 2014] at 1536×1536. As shown in table 3, SPLIT stays in the same high-accuracy regime as the fixed-grid baseline and APT while delivering the best throughput across both model scales. Notably, this throughput gain is achieved even when SPLIT and APT operate at a similar token-reduction level (about 30%): our runtime breakdown confirms that the remaining gap comes from lower decision making overhead in SPLIT, whereas APT incurs higher cost from entropy-based multi-scale scoring when extended beyond two patch sizes. These results indicate that input-level learned policy remains effective for high-resolution dense detection with window attention.

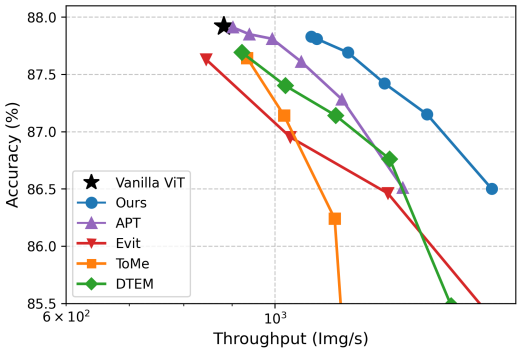


Figure 3: Comparison on ViT-L/14 at 224×224 resolution.

[1] We provide a detailed discussion on training cost in Appendix section E

Model	Res	Img/s	Speedup	mAP	AP50
EVA-02-B	1536	3.66	-	58.93	77.85
Resizing	1536	4.14	+13%	58.43	77.22
APT	1536	4.14	+13%	58.79	77.65
SPLIT (Ours)	1536	4.23	+16%	58.84	77.80
EVA-02-L	1536	1.61	-	62.28	80.80
Resizing	1536	1.91	+19%	61.75	80.27
APT	1536	1.91	+19%	62.07	80.64
SPLIT (Ours)	1536	2.02	+25%	62.24	80.78

Table 3: **Transfer to Object Detection.** SPLIT can be scaled to high-resolution dense image tasks supporting window attention.

Model	Res	Img/s	Speedup	aAcc	mIoU
EVA-02-L	512	6.48	-	86.67	59.77
Resizing	512	7.19	+11%	86.09	58.81
APT	512	7.19	+11%	86.68	59.70
SPLIT (Ours)	512	7.43	+15%	86.53	59.65
EVA-02-L	640	3.44	-	86.83	60.05
Resizing	640	3.82	+11%	86.06	58.83
APT	640	3.82	+11%	86.82	60.01
SPLIT (Ours)	640	3.97	+15%	86.75	59.99

Table 4: **Transfer to Semantic Segmentation.** SPLIT can handle pixel-level fine-grained tasks without compromising visual acuity.

Model	Img/s	Speedup	VQA^{v_2}	GQA	SQA^{I}	VQA^{T}	POPE	MME	MMB	MMB^{C}	MMV
LLaVA-1.5-7B	3.70	-	78.5	61.0	67.8	58.2	86.9	1510.1	64.6	58.1	30.7
Resizing	4.51	+22%	77.5	61.1	66.8	56.5	86.6	1473.8	63.2	58.1	30.2
APT	4.51	+22%	77.9	61.4	67.5	56.9	86.4	1474.0	63.8	58.2	30.8
SPLIT (Ours)	4.37	+18%	78.6	61.7	69.0	58.1	86.6	1497.3	65.2	58.9	31.3
LLaVA-1.5-13B	2.22	-	80.0	63.2	72.7	61.2	87.1	1530.6	68.5	63.4	35.4
Resizing	2.72	+23%	78.9	61.1	72.0	59.1	86.8	1496.9	65.8	62.5	33.9
APT	2.72	+23%	79.4	63.0	72.4	59.5	87.2	1511.2	66.5	63.7	34.7
SPLIT (Ours)	2.66	+20%	79.6	63.1	73.4	60.2	86.9	1504.9	67.5	64.0	36.3

Table 5: **Transfer to VQA**. SPLIT enables significant throughput increase while matching or exceeding performance to the baseline.

Semantic segmentation. We further evaluate SPLIT on semantic segmentation, which requires fine-grained boundary understanding in addition to region-level consistency. Following the EVA-02 segmentation protocol [Fang et al., 2024], we use an UperNet decoder [Xiao et al., 2018] and report results on ADE20K [Zhou et al., 2019, 2017]. table 4 shows that SPLIT closely matches baseline/APT quality on both aAcc and mIoU while consistently improving throughput. In this task, the speed gain aligns with stronger sequence reduction: SPLIT uses roughly 35% fewer tokens, compared with about 30% for APT, yielding additional runtime savings without materially affecting segmentation quality. Together, these results show that SPLIT preserves pixel-level performance while improving efficiency.

Visual question answering. We further evaluate the transferability of SPLIT to multimodal visual question answering by applying it to the vision backbone of LLaVA-1.5 [Liu et al., 2024a]. Following APT [Choudhury et al., 2026], we replace the original uniform patch embedding with our learnable adaptive patchification module, keeping the language model frozen while training the decision module and the vision-language projection layer. As shown in table 5, SPLIT consistently improves inference throughput over vanilla LLaVA while matching or exceeding the baseline accuracy across a wide range of VQA benchmarks [Goyal et al., 2017, Hudson and Manning, 2019, Lu et al., 2022, Singh et al., 2019, Schwenk et al., 2022, Fu et al., 2024, Liu et al., 2024b, Yu et al., 2023]. Although APT achieves a slightly higher speedup at the same target token fraction, it falls short of the baseline on most metrics, whereas SPLIT recovers or surpasses the baseline on nearly every benchmark for both the 7B and 13B variants.

4.3 Ablation

Measuring SPLIT overhead. We next measure the computational overhead introduced by SPLIT. Adaptive patchification breaks the constant-input-shape assumption that GPUs are heavily optimized for, and the decision network itself adds a small forward pass on top of patch embedding. Both costs need to be accounted for before any throughput claims can be trusted. We report this analysis in fig. 4. With no sequence reduction applied, SPLIT incurs a minor 3% overhead relative to the vanilla ViT baseline due to multi-scale patch embedding computations, decision network and additional packing logic to join the multi-scale patch embeddings. Once sequence reduction is enabled, the savings from processing fewer tokens substantially outweigh the overhead.

Lagrange Multiplier Control. As discussed in Section 3.3, the multiplier λ decides the relative weighting of the performance cost and compute cost. In figure 6 we show that a continuously updated λ actively guides the model towards the specified token budget. Using a fixed value of λ for training involves laborious tuning of the value of λ to achieve the desired token budget. We also observe that even though we can achieve similar % tokens using fixed $\lambda = 0.01$ and adaptive λ, the gap in control error between train and val sets is much smaller when using an adaptive λ indicating that it is a better control mechanic.

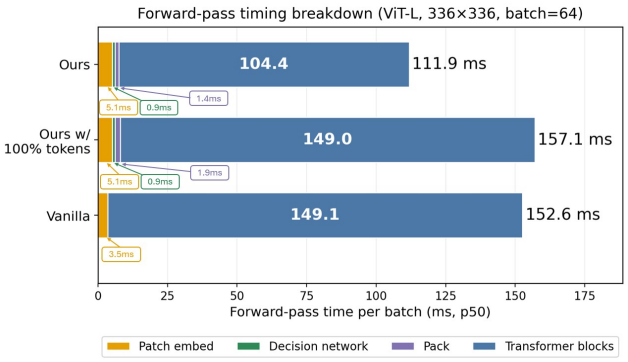
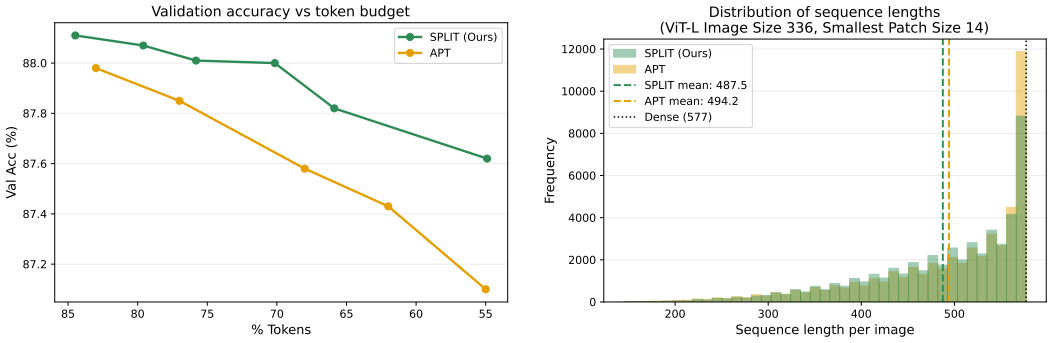


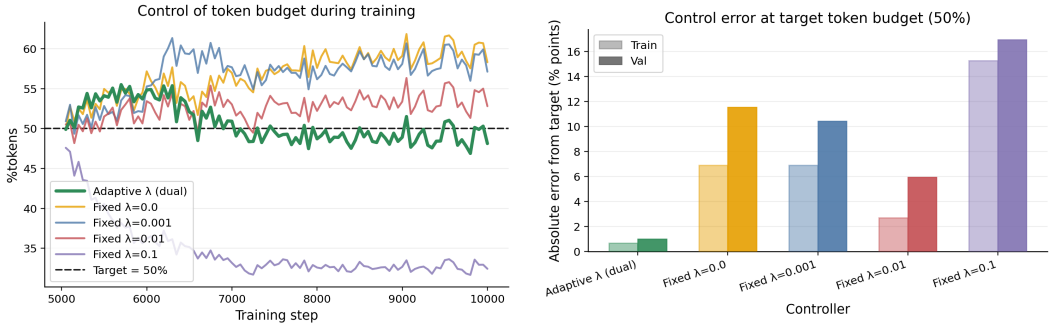
Figure 4: **Forward-pass Timing Breakdown.** SPLIT incurs minimal overhead from decision network and patch splitting operations while saving on transformer blocks, thus resulting in a significant throughput increase



(a) Accuracy vs. % tokens for ViT-L 336/14

(b) Sequence length distribution at same mean % tokens

Figure 5: Comparison of our method against APT.



(a) Token-budget dynamics during training.

(b) Train and validation control error.

Figure 6: Adaptive λ more reliably controls the token budget than fixed-λ baselines. Left: percent tokens used during training, with the dashed line indicating the 50% target. Right: absolute deviation from the target token budget on train and validation sets.

4.4 Comparison against APT

We compare SPLIT directly against APT under matched conditions to isolate the contribution of our decision network. Since both methods use the same multi-scale patch embedding scheme, any performance gap must come from where each method chooses to allocate fine tokens. fig. 5a shows the Pareto frontier of accuracy against token retention on ViT-L at 336×336. Across every token budget we evaluate, SPLIT achieves higher accuracy than APT, with the gap widening as the token budget tightens.



Figure 7: **Patch allocation comparison.** APT (middle) over-tokenizes high-entropy textures regardless of their semantic content, while SPLIT (bottom) concentrates fine tokens around objects.

fig. 5b examines this allocation behavior more closely by plotting the per-image sequence length distributions of the two methods, with their mean sequence lengths matched. Both distributions are skewed toward maximum-length sequences, indicating that a substantial fraction of images are split fully, but APT does this noticeably more often than SPLIT. The mass that SPLIT pulls away from full splits is redistributed toward lower sequence lengths, meaning SPLIT identifies more images for which fewer tokens suffice. APT's tendency to over-tokenize is consistent with its reliance on pixel entropy: visually busy images trigger fine splits regardless of whether the busy regions are semantically informative.

fig. 7 illustrates this difference qualitatively. SPLIT concentrates fine tokens around object boundaries and semantically meaningful structure, while leaving textured but task-irrelevant regions such as patterned walls, tiled floors and repeated backgrounds, at coarse resolution. In contrast, APT frequently allocates fine tokens to these high-entropy but semantically empty regions, consuming budget that SPLIT redirects to objects. Together, these results support our central claim: pixel-level heuristics mis-locate tokens in predictable ways, and a learned decision policy trained against the downstream objective improves patch size allocations.

Quantifying semantic alignment. To make this comparison concrete, we measure *fine-token recall* on ADE20K. We define object recall as the fraction of object-class patches that receive a fine (16×16) token, and boundary recall as the fraction of class-boundary patches that receive a fine token, both at a matched token budget (full definition in Appendix section F). As table 6 shows, SPLIT routes fine tokens to a much larger share of task-relevant patches than the entropy heuristic.

Table 6: **Fine-token recall on ADE20K** at a matched token budget (~ 0.7). Object and boundary recall measure the fraction of object-class and class-boundary patches, respectively, that receive a fine token.

Method	% Tokens	Object recall	Boundary recall
Random	0.700	0.697	0.698
APT (entropy)	0.719	0.808	0.833
SPLIT (Ours)	0.714	**0.907**	**0.945**

5 Conclusion

We presented **SPLIT**, an adaptive patchification framework that learns a content-based patch splitting policy. By treating coarse tokens as lightweight semantic anchors and selectively refining only the regions that matter for the downstream task, SPLIT produces shorter input sequences that are better matched to task-relevant content. We formulate patch refinement as a budget-constrained decision problem and train the splitting policy with reinforcement learning, allowing the model to learn where finer tokens are most useful under a fixed sequence-length budget.

Across ImageNet classification, COCO detection, and ADE20K segmentation, SPLIT matches or surpasses the accuracy of full-resolution ViT baselines while delivering consistent throughput gains over prior adaptive patchification methods. These results show that grounding patch splitting decisions in the model's interpretation of task relevant information generalises well beyond classification and offers a practical path toward more efficient ViTs without sacrificing task performance.

Limitations. SPLIT delivers significant inference speedups by reducing the average token sequence length to a target reduction ratio that is fixed at training time. Evaluating SPLIT for a different desired speedup requires a short retraining run rather than a free inference-time hyperparameter. Conditioning the patchification policy on the desired ratio at inference would remove this constraint. Our second limitation is scope: we evaluate SPLIT only on image understanding tasks. Image generation typically operates at much higher resolutions and with much larger models, a regime where our method's speedup is most pronounced. This remains untested, and we view extending SPLIT to generative settings as a promising direction for future work.

Future Work. Beyond the directions discussed in the limitations, using SPLIT for self-supervised tasks to create policies reusable across multiple tasks is a promising direction. On a similar note, a thorough analysis on the behavior of SPLIT trained on a task and re-used on another is yet to be characterized and may be useful in understanding how to develop a task-agnostic tokenization policy.

Acknowledgments and Disclosure of Funding

This work was supported by Fujitsu Research of America. We thank Souraja Kundu for helpful feedback and assistance with the writing of this paper.

References

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, 2020.

Mengzhao Chen, Mingbao Lin, Ke Li, Yunhang Shen, Yongjian Wu, Fei Chao, and Rongrong Ji. Cf-vit: A general coarse-to-fine method for vision transformer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 7042–7052, 2023a.

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*, pages 17164–17174, 2023b.

Rohan Choudhury, JungEun Kim, Jinhyung Park, Eunho Yang, László A. Jeni, and Kris M. Kitani. Accelerating vision transformers with adaptive patch sizes. In *International Conference on Learning Representations*, 2026.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe RLHF: Safe reinforcement learning from human feedback. In *International Conference on Learning Representations (ICLR)*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image and Vision Computing*, 149:105171, 2024.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. Mme: A comprehensive evaluation benchmark for multimodal large language models, 2024. URL `https://arxiv.org/abs/2306.13394`.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.

Jakob Drachmann Havtorn, Amélie Royer, Tijmen Blankevoort, and Babak Ehteshami Bejnordi. Msvit: Dynamic mixed-scale tokenization for vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 838–848, 2023.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling. *arXiv preprint arXiv:2507.07955*, 2025.

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. Spvit: Enabling faster vision transformers via latency-aware soft token pruning. In *European conference on computer vision*, pages 620–640. Springer, 2022.

Dong Hoon Lee and Seunghoon Hong. Learning to merge tokens via decoupled embedding for efficient vision transformers. *Advances in Neural Information Processing Systems*, 37:54079–54104, 2024.

Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European conference on computer vision*, pages 280–296. Springer, 2022.

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.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024a.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pages 216–233. Springer, 2024b.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12309–12318, 2022.

Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. *Advances in neural information processing systems*, 34:24898–24911, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

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*, 2021.

Tomer Ronen, Omer Levy, and Avram Golbert. Vision transformers with mixed-resolution tokenization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4613–4622, 2023.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge. In *European conference on computer vision*, pages 146–162. Springer, 2022.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019.

Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *IEEE/CVF International Conference on Computer Vision*, 2021.

Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12165–12174, 2022.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Ross Wightman. Pytorch image models, 2019. URL `https://github.com/rwightman/pytorch-image-models`.

Xinjian Wu, Fanhu Zeng, Xiudong Wang, and Xinghao Chen. Ppt: Token pruning and pooling for efficient vision transformers. *arXiv preprint arXiv:2310.01812*, 2023.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018.

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*, pages 10809–10818, 2022.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2023.

Tongtian Yue, Longteng Guo, Jie Cheng, Xuange Gao, Huang, and Jing Liu. Ada-K routing: Boosting the efficiency of MoE-based LLMs. In *International Conference on Learning Representations (ICLR)*, 2025.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.

Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2019.

A Implementation Details

Hardware Setup. All inference-time results, including throughput analysis, were computed on a single NVIDIA A100. ImageNet classification training was conducted on a single node of 8x NVIDIA A100s. Object detection, semantic segmentation, and visual QA experiments were conducted on a single node of 8x NVIDIA H200s. We used a single node for all work in this paper.

Layer-level Merging Baselines. For comparisons against layer-level token reduction methods, we used the official repositories for EViT [Liang et al., 2022], ToMe [Bolya et al., 2023], and DTEM [Lee and Hong, 2024]. We kept all experimental settings (training schedule, optimizers, input resolutions, and other hyperparameters) identical to the original baselines to ensure a fair comparison.

Image Classification. We implemented image classification models using the `timm` library [Wightman, 2019], leveraging its pretrained checkpoints. The ImageNet-1K dataset [Deng et al., 2009] was used for training and evaluation, following prior work [Havtorn et al., 2023, Ronen et al., 2023]. We perform 1 epoch of warmstart and 1 epoch of RL fine-tuning for all experiments. No augmentations are used. All training was conducted on 8 GPUs with a batch size of 512. For the decision network, we used a warm-start learning rate of $1\mathrm{e}{-3}$, a PPO learning rate of $1\mathrm{e}{-5}$, a Lagrangian dual learning rate of $5\mathrm{e}{-4}$ and a warm start threshold update rate of $\eta_\tau = 1\mathrm{e}{-3}$. All image classification experiments involve a single stage of splitting.

Visual QA. For our Visual Question Answering (VQA) experiments, we utilized the official LLaVA-1.5 [Liu et al., 2024a] implementation and its pretrained checkpoints. Unlike the original approach, which fine-tunes on the entire dataset for one epoch, we fine-tuned only 5% of the dataset and do one epoch for warmstart and one epoch for RL. To adapt to this setting, we reduced the learning rate by a factor of 10 while following all other fine-tuning procedures recommended by LLaVA. The base image resolution was set to 336 with a patch size of 14, as specified in LLaVA's default configuration. We keep our retained token fraction to 0.7 and 0.001 for warm start threshold update rate.

Object Detection. For object detection, we used the official implementation of EVA-02 [Fang et al., 2024] along with its pretrained checkpoints, which utilize a window attention mechanism. Fine-tuning was conducted following the recommended procedures outlined in EVA-02. Consistent with our previous experiments, we fine-tuned for 5% of the total iterations, split evenly between the warm-start and PPO phases, while reducing the learning rate by a factor of 10. Following EVA-02's settings, the image resolution was 1536 with a base patch size of 16. SPLIT additionally produces patches at sizes 128, 64, 32, and 16. For the decision network, we used a warm-start learning rate of $1\mathrm{e}{-3}$, a PPO learning rate of $1\mathrm{e}{-5}$, and a Lagrangian dual learning rate of $5\mathrm{e}{-3}$, with a target retained token fraction of $\rho = 0.7$ and a warm start threshold update rate of $\eta_\tau = 1\mathrm{e}{-3}$.

Semantic Segmentation. We also utilized the official EVA-02 implementation along with its pretrained checkpoints for semantic segmentation. The ADE20K dataset [Zhou et al., 2019, 2017] was used for training and evaluation. Fine-tuning followed the recommended procedures outlined in EVA-02. In alignment with our previous experiments, we fine-tuned for 5% of the total iterations, split evenly between the warm-start and PPO phases, while reducing the learning rate by a factor of 10. According to EVA-02's settings, the image resolution was either 512 or 640, with a base patch size of 16. SPLIT produces patches at sizes 64, 32, and 16. The decision network hyperparameters match the detection setting, except that the target retained token fraction is set to $\rho = 0.65$.

B Choice of Policy Gradients over Differentiable Relaxations

In this section, we provide more detail on our decision to use policy gradients which shape our core algorithm. As discussed in Section 3.3, training the decision network requires gradients to flow through an indexing operation that selects whether a patch is represented by a coarse token or by 4 fine tokens in the final sequence passed to the transformer backbone. This operation is non-differentiable as the decision network makes a binary decision and that decision determines which tokens exist in the sequence. Using a differentiable relaxation possible in principle, MSViT [Havtorn et al., 2023] builds one for mixed-scale tokenization. However, since the difficulty is that our discrete decision is structural, a relaxation then has only two options:

Option 1: Keep both scales. A soft mixture, or a masked union as in DynamicViT [Rao et al., 2021], lets gradients flow, but every scale must be materialized. For a single splitting stage, this means materializing $5N$ tokens (N coarse + $4N$ fine), versus the $4N$ of the dense sequence. With attention cost quadratic in sequence length and activation memory growing with it, this becomes prohibitive at 1536×1536 resolution. Hierarchical splitting makes this worse: a fine token exists only if every ancestor split, so its relaxed weight is a product of gates across levels and gradients to each gate are filtered through the approximations of the others.

Option 2: Compute only the selected scale. Drop the un-selected tokens to keep the training costs from increasing beyond the dense sequence limit and use Straight-Through Estimators for the binary decision. In this case the gradients are biased and the unselected tokens produce no learning signal and the policy can never directly learn whether a coarse token would have sufficed and must rely on stochastic exploration for this.

A differentiable relaxation produces a biased gradient estimator and must choose between materializing sequences longer than dense sequences or losing signal on the unselected branch of the decision. Policy gradients avoid both problems and trains on exactly the hard, reduced sequences used at inference, with unbiased gradients. Ada-K routing [Yue et al., 2025] chose PPO for the same structural problem in MoE routing, stating "direct optimization ... is precluded due to the non-differentiable nature of the sampling operation". Also, methods where differentiable gating was tried needed extra stabilizers: MSViT's batch-shaping loss against gate collapse, DynamicViT's distillation loss.

What the RL formulation contributes. The RL phase of training using policy gradients on hard sequences yields measurable gains. Table 9 shows the PPO phase improves object detection mAP monotonically over a warm-start-only policy ($58.06 \rightarrow 58.84$ at a matched budget). Table 14 shows that it sharpens semantic allocation by $+2.0$ pp thing and $+1.1$ pp boundary recall. Also, since our policy gradients allows our training to use hard sampled masks, λ regulates the actual deployed token count rather than a soft proxy (Figure 6).

C Sensitivity to Hyper-parameters

A natural concern for an RL-based method is how sensitive it is to training hyper-parameters, and whether the added optimization introduces brittle tuning. To address this, we sweep each of SPLIT's PPO-specific and warm-start hyper-parameters, varying one at a time around the default configuration used in the main paper while holding all others fixed.

PPO hyper-parameter sweeps Table 7 varies the PPO clip ratio ϵ, the number of rollouts K, the number of PPO epochs, the warm-start balancer rate, and the backbone learning rate during the warm-start. The results in Table 7 correspond to the image classification task using the configuration: ViT-L, 224×224, 0.5 target token fraction. We observe that the accuracy is within $\pm 0.2\%$ across all settings except in the case of (i) running ≥ 4 PPO epochs over the same rollouts and (ii) backbone learning rates above $1\text{e}{-6}$ degrade the pretrained features (85.73% at $1\text{e}{-5}$). Both failure modes have well-understood causes: (i) over-optimizing the PPO objective and (ii) over-writing pretrained features. These are avoidable pitfalls and do not reflect instability unique to SPLIT.

Table 7: Single hyper-parameter sweeps on ImageNet-1K (ViT-L, 224×224, target token fraction 0.5). A run using the default hyper-parameters scores 86.49 at token fraction 0.586.

Hyper-parameter (default)	**Values swept**	**Test acc (%)**
PPO clip ratio ϵ (0.10)	0.05 / 0.10 / 0.20	86.49 / 86.49 / 86.48
Rollouts K (4)	2 / 4 / 6	86.49 / 86.49 / 86.33
PPO epochs (2)	1 / 2 / 4	86.50 / 86.49 / 83.28
Warm-start balancer rate (0.005)	0.001 / 0.005 / 0.02	86.44 / 86.49 / 86.54
Backbone LR ($1\text{e}{-6}$)	$5\text{e}{-7}$ / $1\text{e}{-6}$ / $5\text{e}{-6}$ / $1\text{e}{-5}$	86.55 / 86.49 / 85.97 / 85.73

Dual rate and policy learning rate are coupled. The Lagrangian dual rate η_λ controls how quickly the multiplier λ reacts to token-budget violations and the policy learning rate controls how fast the

policy changes based on the combined cost weighted by λ. Due to this coupling of the two learning rates through λ, it makes sense to observe their joint behavior, which we present in Table 8 using the same configuration as Table 7 with a target token fraction of 0.5 and varying only the two learning rate hyper-parameters.

We observe that an under-reactive dual ($\eta_\lambda = 1\text{e}{-4}$) overshoots the target budget by 22–34% as the cost does not adjust fast enough to enforce the budget. For $\eta_\lambda \geq 5\text{e}{-4}$ the two hyper-parameters substitute for one another, anti-diagonal cells match within 0.1% and effectively behave as a single degree of freedom.

Table 8: Joint sweep of the Lagrangian dual rate η_λ and the RL-phase policy learning rate (ViT-L, 224×224, target token fraction 0.5). Each cell reports test accuracy (%) / realized token fraction. Defaults (†) are $\eta_\lambda = 5\text{e}{-4}$ and policy LR $= 5\text{e}{-6}$.

	Policy learning rate		
$\boldsymbol{\eta_\lambda}$	$1\text{e}{-6}$	$5\text{e}{-6}^\dagger$	$1\text{e}{-5}$
$1\text{e}{-4}$	85.2 / 0.72	86.3 / 0.81	86.2 / 0.84
$5\text{e}{-4}^\dagger$	86.1 / 0.62	86.4 / 0.58	86.1 / 0.51
$1\text{e}{-3}$	86.3 / 0.55	86.1 / 0.52	86.1 / 0.51

Number of warm-start and RL iterations. We also study how performance depends on the iterations used during the warm-start and RL phases of training through two complementary experiments. In the first experiment, we fix a 1500-iteration warm-start on object detection and track performance across subsequent RL iterations. Table 9 shows that mAP increases monotonically with RL iterations ($58.06 \rightarrow 58.84$). The PPO phase contributes $+0.8$ mAP over the warm-start decisions at a matched token budget. In the second, we vary the split of iterations between the two phases on semantic segmentation while holding the *total* iteration count fixed at 3000. As Table 10 shows, final mIoU stays within 0.05 mIoU across all three splits and during training we observe that shorter warm-starts only enlarge the transient token-fraction oscillation early in RL. The warm-start therefore mainly stabilizes the beginning of RL rather than setting final quality.

Table 9: RL-iterations ablation on object detection (EVA-02-B @ 1536, $\rho = 0.70$) with a fixed 1500-iteration warm-start. Default configuration marked †.

RL iterations	0 (warm-start only)	500	1000	$1500^\dagger$
COCO mAP	58.06	58.20	58.52	58.84

Table 10: Warm-start/RL iterations split on segmentation (EVA-02-L @ 640, ADE20K, $\rho = 0.65$) with the total iteration count fixed at 3000. Default configuration marked †.

Warm-start iters	**RL iters**	**Total iters**	**Final mIoU**
1000	2000	3000	59.34
$1500^\dagger$	$1500^\dagger$	3000	59.39
2000	1000	3000	59.38

D Robustness to Distribution Shift

A desirable property of any adaptive tokenization method is that it must be robust to deviations from the data distribution on which the tokenization was trained or tuned. We run experiments to observe how the trained policy reacts to (i) change in distribution of test data and (ii) corruption of input images. We compare the behavior of SPLIT against APT [Choudhury et al., 2026] to show the benefit of using a learnable approach as compared to a heuristic driven approach.

Distribution shift. We applied the ImageNet-trained policy to five datasets at 224×224 without retraining the policy: CIFAR-100, ImageNet-V2, -Sketch, -R, -A. For ImageNet $\rightarrow$ CIFAR-100 (resized), we learn a linear probe fit with the backbone frozen, since the label space of CIFAR-100 is different from that of ImageNet. For the four ImageNet variants, we use the same 1000-class head (frozen); specifically ImageNet-R/A are standard 200-class subsets, and are scored by restricting the logits to the subset. We compare with APT in two forms: *fixed-τ* , a direct transfer condition, since recalibration needs target-domain data, and τ *recalibrated* per dataset to SPLIT's token fraction for a matched-budget accuracy comparison.

We run the experiments in Table 11 with a target token fraction of 0.85 while training. SPLIT maintains a token fraction within $0.74 - 0.90$ across all test datasets and is at most ~ 0.11 from its training target. Fixed-τ APT swings over $0.50 - 0.89$, overspending on CIFAR-100, collapsing to 0.50 on ImageNet-Sketch. Even against per-dataset recalibrated APT, SPLIT is more accurate on the four ImageNet variants by $+0.14$ to $+0.77$, while staying within 0.1 pp of the dense baseline.

Table 11: Transfer of the ImageNet-trained SPLIT policy to five datasets with no retraining (ViT-L, 224×224; training target token fraction 0.85). For SPLIT and fixed-τ APT we report accuracy with the realized token fraction in parentheses. "APT recal." tunes τ per dataset to SPLIT's token fraction.

Dataset	**Dense**	**SPLIT (tok. frac.)**	**APT fixed-τ (tok. frac.)**	**APT recal.**
CIFAR-100 (probe)	89.87	89.26 (0.795)	89.60 (0.831)	89.49
ImageNet-V2	79.13	79.07 (0.90)	78.93 (0.89)	78.93
ImageNet-Sketch	61.14	61.22 (0.74)	56.82 (0.50)	60.73
ImageNet-R	77.19	77.06 (0.83)	75.71 (0.74)	76.42
ImageNet-A	64.45	64.37 (0.90)	63.28 (0.89)	63.60

Input corruption. To isolate the behavior in a controlled setting, we add Gaussian pixel noise of increasing standard deviation σ to the ImageNet test data and evaluate at 336×336 (ViT-L). In Table 12 we start with SPLIT and APT having the same token fraction under zero noise. As noise increases the entropy of every patch, APT progressively splits more patches and its token fraction climbs from 0.53 to 0.95, while SPLIT holds its budget ($0.53 \rightarrow 0.59$).

Table 12: Gaussian pixel noise of standard deviation σ added to ImageNet test data (ViT-L @ 336×336). Each entry is test accuracy (%) at the realized token fraction. APT τ is calibrated to match SPLIT's zero-noise token fraction.

Noise σ	**SPLIT acc @ tok. frac.**	**APT acc @ tok. frac.**
0.00	87.7 @ 0.53	87.5 @ 0.53
0.05	86.5 @ 0.53	86.7 @ 0.62
0.10	84.9 @ 0.54	85.3 @ 0.75
0.15	83.0 @ 0.56	82.8 @ 0.92
0.20	80.1 @ 0.59	79.3 @ 0.95

E Training Costs

We quantify SPLIT's training overhead for EVA-02-L at 640×640 on ADE20K, with two splitting stages (three patch scales). Table 13 uses identical hardware (single A100 GPU) and batch size, and a matched 3000-iteration schedule for all rows and reports per-iteration and per-image latency alongside total wall-clock training time. Even though SPLIT has two phases of training (warmstart and PPO), the training budget is *matched, not doubled*: SPLIT partitions APT's 3000-iteration schedule into 1500 warm-start and 1500 PPO iterations. Because the backbone is frozen throughout the PPO phase, the backbone in SPLIT receives *half* the updates APT does. In total, SPLIT trains only about 12 minutes longer than APT, an overhead that comes almost entirely from the PPO phase.

It is important to note that all methods, including heuristic driven ones like APT would require updating the backbone to adapt to the change in token distribution with a new tokenization scheme. The warm-start phase plays exactly this role for SPLIT, and in no configuration does SPLIT's

Table 13: Training cost on ADE20K (EVA-02-L @ 640, two splitting stages, 1×A100 GPU).

Config	ms/iter	ms/img	Iters	Total
Vanilla (dense)	963	60	3000	48.2 min
APT finetune	809	51	3000	40.45 min
SPLIT warm-start	910	57	1500	22.8 min
SPLIT PPO phase	1186	74	1500	29.6 min
SPLIT total	—	—	3000	52.4 min

backbone receive more updates than APT's. On classification, where recipes are epoch-based, the PPO phase adds one extra epoch with the backbone frozen ($\sim$80 minutes total on 2×A100 for the full two-phase run). SPLIT's gains are therefore attributable to policy learning, not additional finetuning.

Beyond training cost, this dependence on updating the backbone also constrains how an adaptive tokenizer can be used at inference. Changing the token budget at run time not only requires a tokenizer that allows for adjusting the budget but also needs the backbone to adapt to the new distribution of tokens for optimal performance. APT allows for budget control by adjusting the threshold τ to change the token budget, but since the backbone was trained with a different budget, the backbone cannot perform optimally unless we separately train a backbone for the new τ. SPLIT in its current formulation does not support changing the token budget without re-training, this can be approximately achieved without training by shifting probabilities of the decision network, but this still does not address the problem of the backbone not being tuned for a new budget. We note this per-budget retraining requirement as a limitation (Section 5) and as a direction for future work. The ideal resolution would train the backbone with a randomly sampled ρ and a tokenizer that accepts ρ as an input on which decisions are conditioned, resulting in a single model taking a desired token fraction directly as input and removing the need for any per-budget adaptation.

F Quantitative Assessment of Semantic Alignment

To quantitatively show that SPLIT's fine-token allocation is semantically aligned, we test whether the finest (16×16) patches land on semantically important regions. We measure fine-token recall on ADE20K `val` against SceneParse150 ground-truth masks (unlabeled / not-in-150 pixels are ignored). For each image we build (1) a binary map of which 16×16 cells the policy keeps at fine resolution, and (2) a patch-level ground-truth label by pixel majority vote per cell; *thing* patches are object-class cells, and *boundary* patches are cells whose majority label differs from that of a 4-neighbor. Thing (resp. boundary) recall is the fraction of such patches that receive a fine token, i.e., of all thing (resp. boundary) patches, what share is allocated fine resolution. We compare Random, APT (entropy), and SPLIT at matched budgets ($\sim$0.7 token fraction) in Table 14.

Table 14: Fine-token recall on ADE20K `val` at a matched token budget ($\sim$0.7). *Thing* recall is the fraction of object-class patches, and *boundary* recall the fraction of class-boundary patches, that receive a fine (16×16) token.

Method	Token fraction	Thing recall	Boundary recall
Random	0.700	0.697	0.698
APT (entropy)	0.719	0.808	0.833
SPLIT, warm-start only	0.708	0.887	0.934
SPLIT, full	0.714	**0.907**	**0.945**

As a sanity check, the Random baseline recalls objects and boundaries at approximately its token budget (0.697 and 0.698 at a 0.700 fraction), confirming that the metric is properly calibrated. At a matched budget, SPLIT allocates fine tokens to a substantially higher share of objects (+9.9 pp over APT) and boundaries (+11.2 pp), and the PPO phase further sharpens alignment beyond the warm-start policy (+2.0 pp thing, +1.1 pp boundary). This quantifies what our qualitative visualizations (Fig 7) shows: the entropy heuristic spends its budget on high-entropy but semantically empty texture, whereas the learned policy reallocates it toward task-relevant structure.

G Decision Network Architecture

We also ablate various different decision network architectures with the same parameter count to justify our choice of the cross attention architecture. In Table 15 we test our choice against using 2 layer MLPs with input as (i) only the embedding of the un-split parent patch or (ii) embedding of the un-split parent concatenated with embedding of all the child patches. We find that using the embeddings of both the parent and child embedding as inputs to the decision network improves performance, and our cross attention design outperforms using simple MLP layers for this.

Decision Network Architecture	**Param Count**	**% Tokens**	**Val Acc (%)**
$\mathrm{MLP}(x_{parent})$	1.115M	89.59	87.69
$\mathrm{MLP}(\mathrm{concat}([x_{parent}, x_{child}^{(1)}, \ldots, x_{child}^{(4)}]))$	1.116M	90.08	87.74
Cross Attention 3.2	1.115M	89.53	**87.81**

Table 15: Comparison of different decision network architectures using ViT-L 224/14

H Qualitative Results

SPLIT generalizes effectively to downstream visual tasks that require spatial precision, including object detection and semantic segmentation. As illustrated in Figure 9 and Figure 10, SPLIT reliably allocates larger patches to uniform background regions while preserving fine-grained structure with smaller patches around object boundaries and detail-heavy areas. The resulting predictions match the vanilla full-resolution baseline closely, supporting accurate bounding box regression and maintaining the pixel-level fidelity required for segmentation. To verify that mixed-resolution tokenization does not distort the model's internal representations, Figure 11 compares backbone feature maps under a shared PCA basis: SPLIT's features exhibit the same spatial structure as the vanilla baseline, indicating that the multi-stage splitting policy preserves the learned representation rather than disrupting it.

Figure 12 illustrates the failure mode that emerges when the token budget is pushed beyond the operating point used in our main experiments. At a 45% retention target, SPLIT begins to under-tokenize genuinely informative regions, producing coarse object boundaries and missing thin structures in the predicted segmentation. These cases mark the boundary of the budget–accuracy trade-off: while SPLIT preserves task performance across the budgets used in our main results, very aggressive budgets force the policy to drop fine tokens in regions where they are needed.

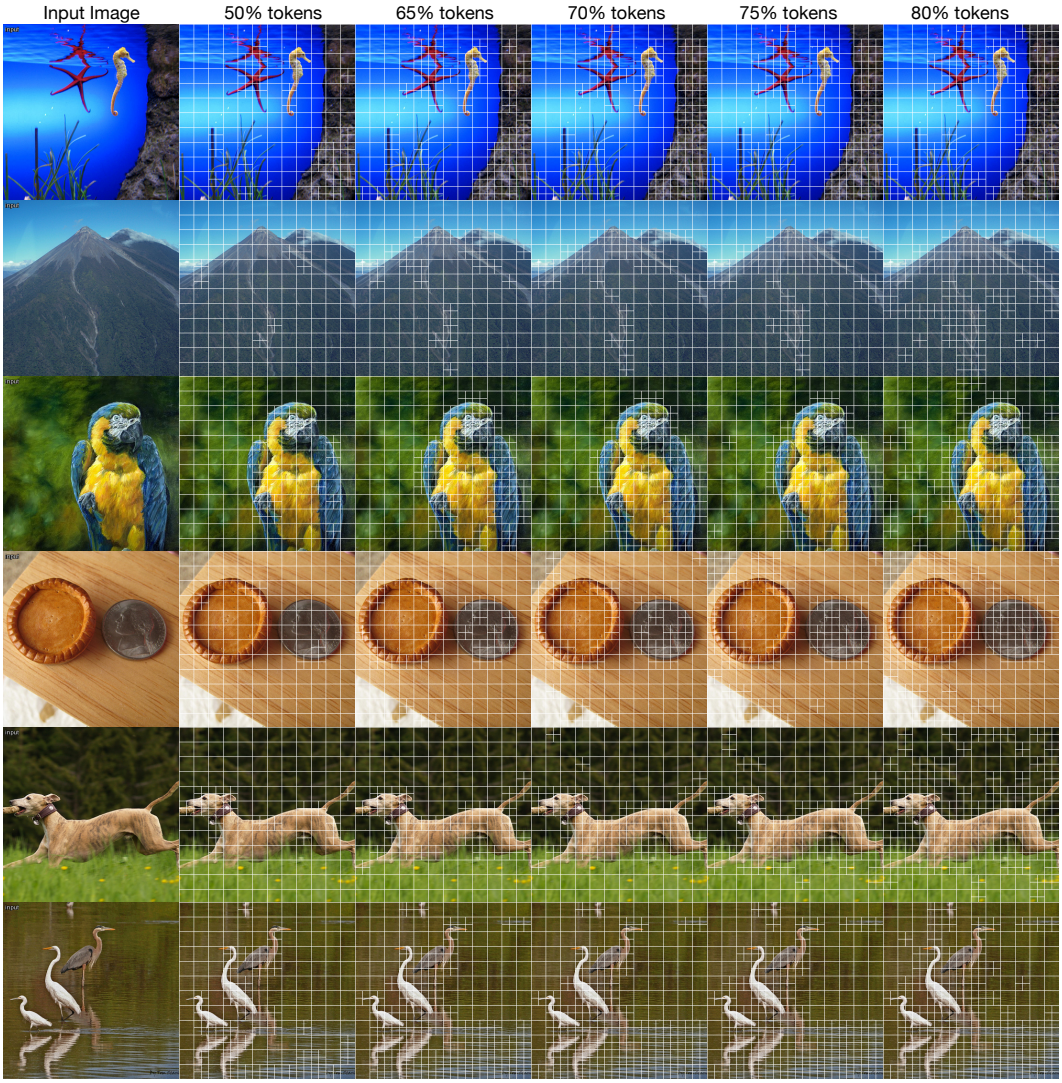


Figure 8: **Splitting decision at different token budgets** Visualizing the patch splitting decisions at various token budgets shows how the model progressively prioritizes patches based on content.



Figure 9: **Object detection qualitative comparison (Vanilla vs Ours).** We compare fixed-grid ViT inference (Vanilla) against our learned multi-stage patch splitting method. For each example, we visualize the patch assignment map and the corresponding detection output. Our method allocates finer patches to detail-heavy regions while preserving detection quality.

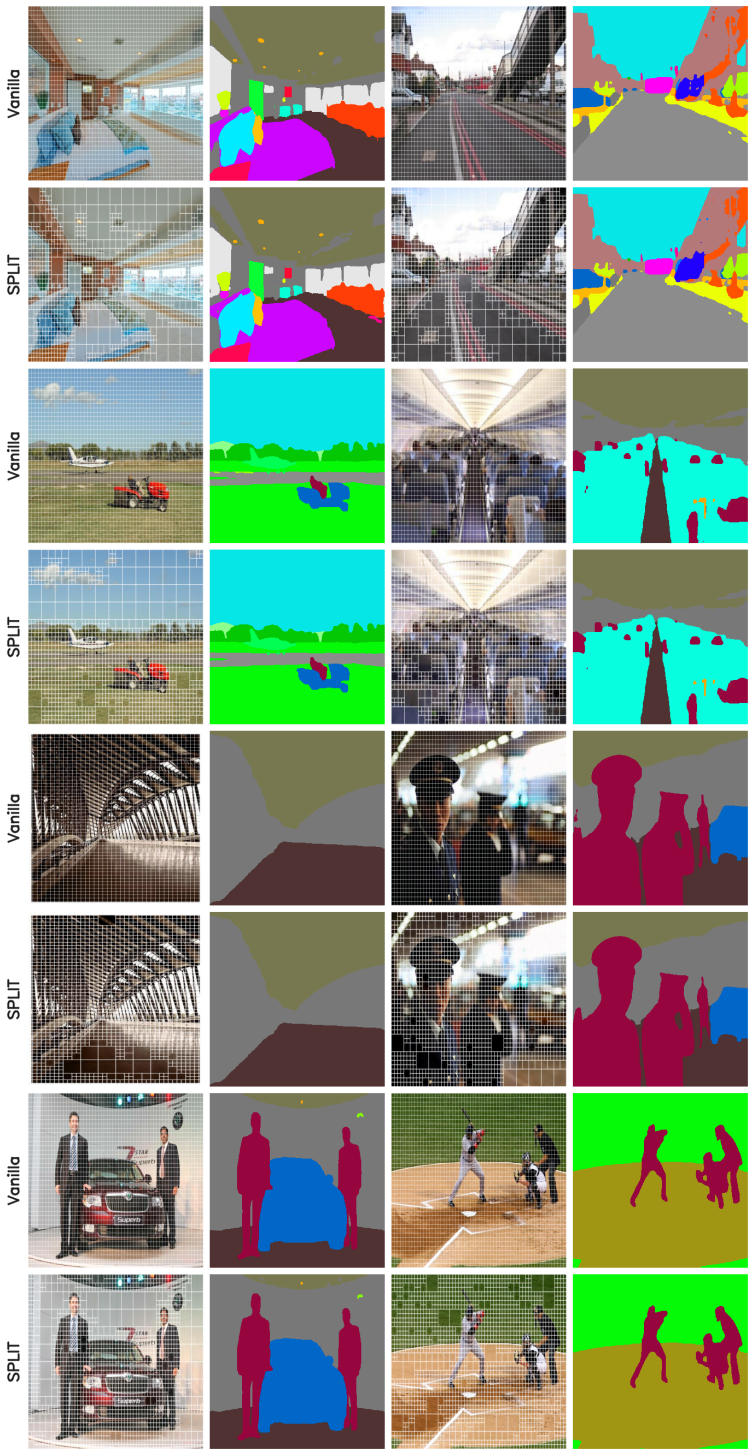


Figure 10: **Semantic segmentation qualitative comparison (Vanilla vs Ours).** We compare fixed-grid ViT inference (Vanilla) against our learned multi-stage patch splitting method. For each example, we visualize the patch assignment map and the corresponding segmentation output. Our method allocates finer patches to detail-heavy regions while preserving segmentation quality.

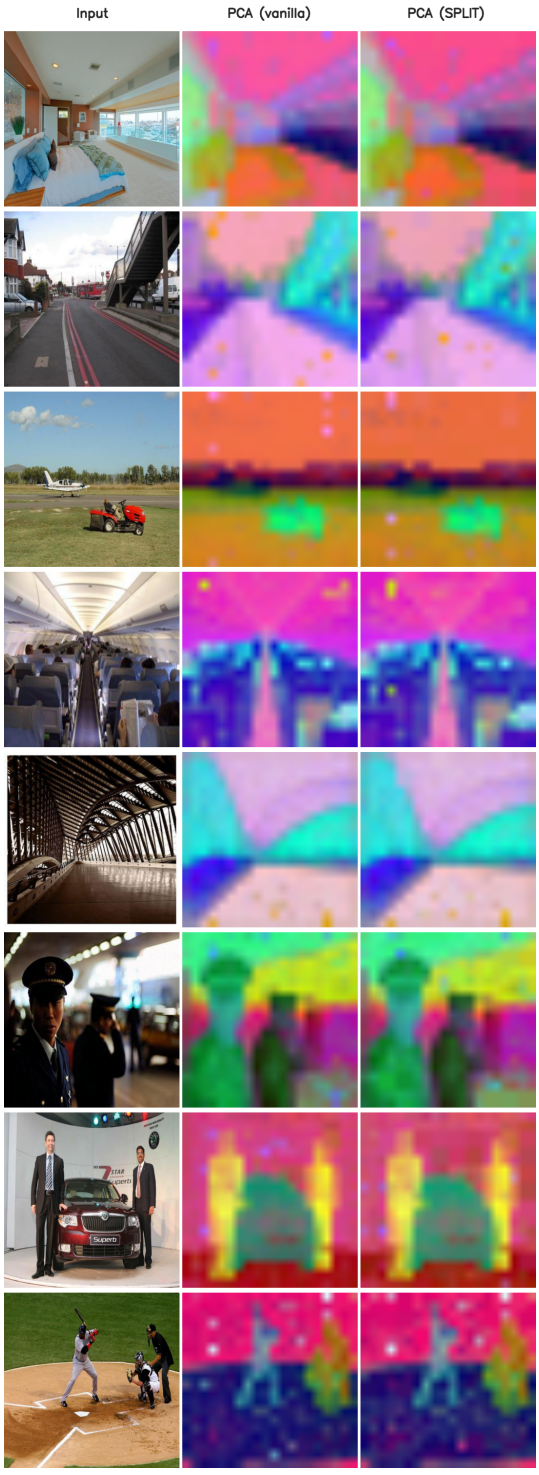


Figure 11: **PCA visualization of backbone feature maps (Vanilla vs Ours).** We project high-dimensional backbone features to RGB using a shared PCA basis fit on Vanilla features, then apply the same projection to our method. Comparable spatial structure in the PCA maps indicates that mixed-resolution splitting does not corrupt the learned representation.

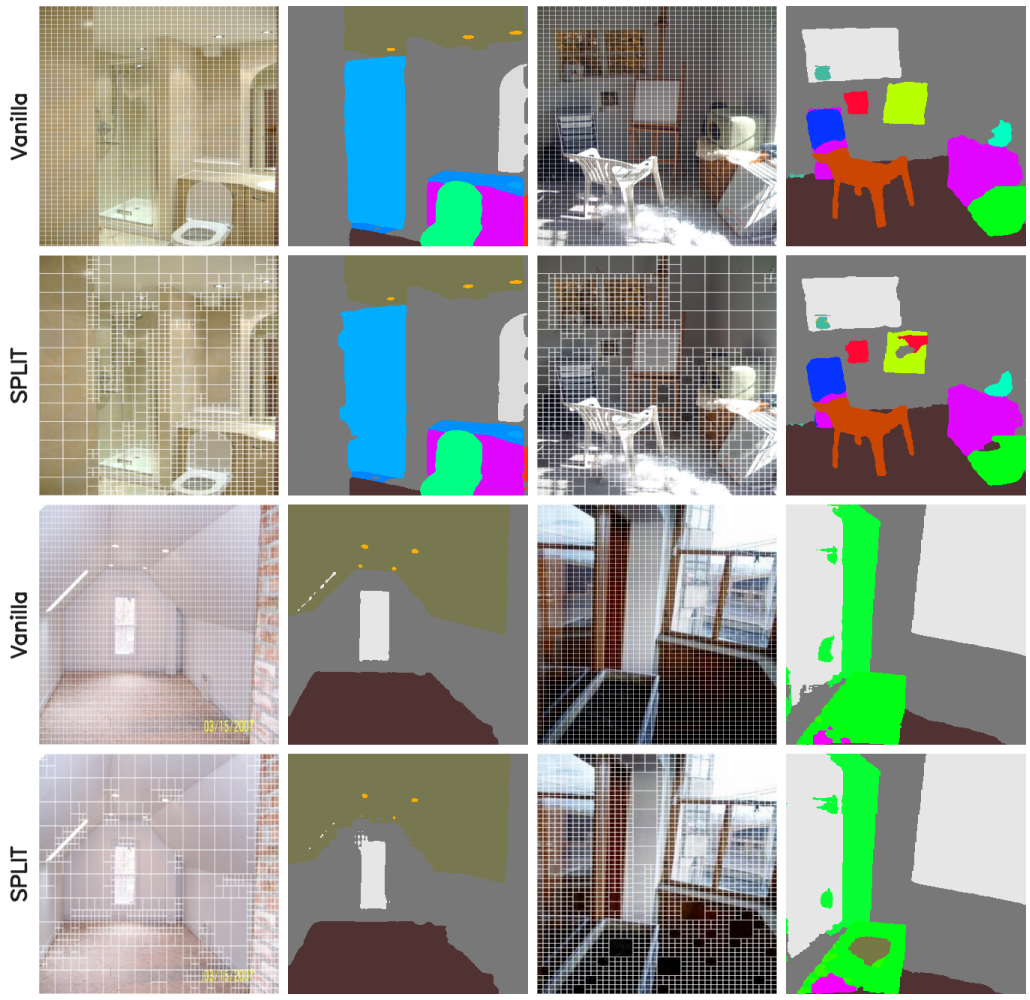


Figure 12: **Failure cases under aggressive token budgets.** When the target sequence length is pushed below the level used in our main experiments (here, 45%), SPLIT begins to under-tokenize semantically important regions, leading to coarse boundaries and missed thin structures in the predicted segmentation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In our paper, we systematically discuss the current limitations of adaptive patch embedding and clearly motivate the need for task-aware patchification. The abstract and introduction accurately summarize our main contributions, including the proposed SPLIT framework, the use of coarse semantic anchors for adaptive patch selection, the reinforcement-learning-based routing policy under a fixed token budget, and the empirical validation across multiple downstream visual tasks.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of our method following the conclusion

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Full training recipe is provided for all results listed in the tables showing the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Our paper fully relies on publicly available data and code, and our own code will be released upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We include all training details and hyperparameter selection in the experiment and supplementary section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Compute Limits.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the detailed information about the computer recources in the supplementary

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: NA

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Our work aims to improve the efficiency of Vision Transformers by reducing redundant visual tokens before full-resolution processing.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: our works poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Our paper are credited to the public available codebase.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce our new assets in our paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.