AHT-ViT: Adaptive Halting Transformer with Planned Depth Execution

Vitalii Shutov, Arip Asadulaev

ITMO University

Abstract. Vision Transformers (ViTs) offer strong performance but face high computational costs from processing all tokens through their full depth. Standard ViTs lack adaptivity. This work introduces Adaptive Halting Transformer (AHT-ViT) to enhance efficiency by dynamically adjusting processing depth per token. AHT-ViT employs hierarchical "planner" modules predicting token-specific target halting depths and an extremely parameter-efficient "supervisor" mechanism (two shared parameters) generating per-layer halting scores. Tokens halt when their cumulative score crosses a threshold. A novel KL divergence-based loss, L_{target_depth}, explicitly aligns executed halting distributions with planned depths. Evaluation on ImageNet, Places365, and CIFAR-100 using DeiT-S shows AHT-ViT achieves an improved accuracy-efficiency trade-off compared to its static baseline and demonstrates competitive performance against other adaptive methods (DynamicViT, A-ViT) evaluated under the same conditions, while significantly reducing FLOPs. Key hyperparameters were selected via grid search on a validation split.

Keywords: Vision Transformer, Adaptive Computation, Early Exit, Dynamic Depth, Model Efficiency, Image Classification.

1 Introduction

Vision Transformers (ViTs) [1], achieving state-of-the-art results across diverse computer vision tasks including classification [2], object detection [22], and segmentation [23], process images as sequences of tokens using self-attention [3]. However, their standard architecture employs a static computational graph, processing every token through the full network depth (L layers). This incurs significant computational overhead [4], independent of the complexity or importance of different image regions, hindering deployment in resource-constrained scenarios.

This motivates research into adaptive computation strategies, surveyed in works like [4,24]. Token pruning or sparsification methods [5,6,25,26] discard or downsample tokens, saving computation but risking information loss detrimental to dense prediction tasks. Token merging [19] offers an alternative way to reduce token count. Orthogonally, adaptive depth processing [7, 8, 18], allows tokens to exit the computation early based on input characteristics. This preserves all spatial tokens while reducing the average computational depth, making it suitable for various downstream tasks.

This paper introduces AHT-ViT (Adaptive Halting Transformer), a novel adaptive depth method featuring a planner-supervisor system. Hierarchical planner modules predict token-specific target halting depths ($N^s_{\text{target},k}$) based on evolving features. An extremely parameter-efficient per-layer supervisor mechanism (using shared γ, β, adding <0.5M parameters) generates halting scores (h^l_k). Tokens halt computation when their cumulative score reaches a threshold T. Our core contribution is the L_{target_depth} loss function, which uses Kullback–Leibler (KL) divergence [9] (a concept also used in models like VAEs [37]) to explicitly align the supervisor's executed halting patterns with the hierarchical planner's predictions. Experiments on standard benchmarks (ImageNet [10], Places365 [15], CIFAR-100 [16]) show AHT-ViT improves the accuracy-FLOPs trade-off over static baselines and performs competitively against other adaptive methods evaluated under identical conditions.

Bibhu Dash et al: EDU, IOTCB, MLDA - 2025
pp. 41-50, 2025. IJCI – 2025

DOI:10.5121/ijci.2025.140404

2 Related Work

The $N \times L$ computational cost of standard ViTs [1], driven by both self-attention ($O(N^2D)$) and MLPs ($O(NLD^2)$), motivates diverse efficiency improvements. Some approaches modify the attention mechanism itself (e.g., Linformer [27], Performer [28]) or build hierarchical structures inspired by CNNs like ResNet [29] or DenseNet [30] (e.g., PVT [31], Swin Transformer [17], CrossViT [32]). Others focus on dynamic computation.

2.1 Token Sparsification

These methods reduce the effective number of tokens N. DynamicViT [5] uses dedicated modules to prune less informative tokens hierarchically. SPViT [6] employs soft pruning based on learnable scores. EViT [25] prunes based on attention scores, while Patch Slimming [26] removes tokens progressively. Token merging [19] fuses similar tokens. While effective at reducing FLOPs, these methods inherently alter the spatial token set.

2.2 Adaptive Depth

These methods reduce the average computation depth $\bar{L}$. Early work includes ACT [7] for RNNs and spatial ACT [8] for CNNs. BranchyNet [18] introduced early exiting classifiers. For Transformers, methods often use confidence scores (e.g., FastBERT [33]) or layer similarity (e.g., DeeBERT [34]) to trigger exits. A-ViT [11] introduced an efficient per-token halting mechanism using a shared-parameter (γ, β) supervisor, guided by a global distributional prior loss (L_{distr}) matching the average halting profile to a target. While parameter-efficient, its global guidance lacks token-specific planning based on evolving hierarchical context. Other related ideas involve learning how many steps to compute, like PonderNet [20]. Conditional computation via Mixture-of-Experts (MoE) [35, 36] dynamically routes tokens to specialized MLPs but typically doesn't alter depth.

AHT-ViT builds upon the efficient supervisor structure of A-ViT but replaces the global prior with hierarchical planners and the L_{target_depth} loss for explicit, per-token planned depth alignment. This distinguishes it from A-ViT's global guidance and DynamicViT's pruning. We evaluate these methods under identical conditions in Section 4.

3 Proposed Method: AHT-ViT

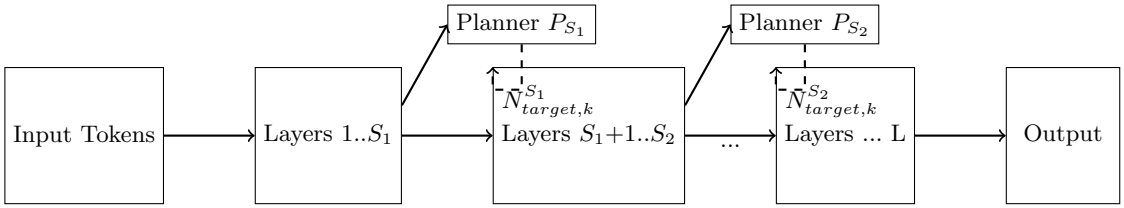
3.1 Architecture Overview

AHT-ViT adds planners and supervisors to a ViT backbone (Fig. 2). Planners at stages $S = \{S_1, \dots\}$ output $N^s_{\text{target},k}$. Supervisors in each layer l output h^l_k. Tokens accumulate scores and halt at layer N_k when the sum reaches T. L_{target_depth} aligns N_k distributions with $N^s_{\text{target},k}$ distributions.

3.2 Hierarchical Planners

Lightweight MLPs inserted after stages $S = \{3, 6, 9\}$. Planner P_s uses token k's features t^s_k (processed local features + aggregated global context features) to predict target depth $N^s_{\text{target},k} \in [1, L]$.

Fig. 1. Example visualizations of AHT-ViT halting behavior. Each panel shows the original image (left) and the halting visualization (right), where overlaid patches indicate tokens halted early. Note how background/uniform areas are often halted, while foreground details remain.



Supervisors generate h_k^l in each layer based on shared γ, β. Planners at stages S_s provide target depths $N^s_{target,k}$. Tokens halt when cumulative score $\geq T$.

Fig. 2. Conceptual overview of AHT-ViT with multiple planner stages $(S_1, S_2, \dots)$.

3.3 Per-Layer Supervisors & Halting

Layer l calculates halting score $h_k^l = \sigma(\gamma \cdot \text{supervisor\_feature}_k^l + \beta)$ using shared scalars γ, β. The $\text{supervisor\_feature}_k^l$ is a single dimension extracted post-FFN-layer1. Cumulative score $C_k^l = C_k^{l-1} + h_k^l$ (with $C_k^0 = 0$). Token k halts at layer $N_k = \min\{l \mid C_k^l \geq T \text{ or } l = L\}$. We use $T = 1.0$, finding performance robust in preliminary tests for $T \in [0.9, 1.1]$. Halted tokens are masked in subsequent computations.

3.4 Training Objective

The composite loss is $L_{total} = L_{task} + \lambda_p L_{ponder} + \lambda_t L_{target\_depth}$.

– **Task loss** L_{task}: the usual cross-entropy between the model logits and ground-truth labels.

Table 1. Notation used throughout the paper.

k	token index $(1 \,..\, K)$
l	layer index $(1 \,..\, L)$
s	planner stage index $(s \in S)$
h_k^l	halting score for token k in layer l
$N^s_{\text{target},k}$	planner-predicted exit layer for token k at stage s
N_k	actual exit layer chosen for token k by supervisor

– **Ponder cost** L_{ponder}: the expected computation budget,

$$L_{ponder} = \frac{1}{K} \sum_{k=1}^{K} N_k, \tag{1}$$

where N_k is the layer in which token k halts.

– **Target-depth alignment** L_{target_depth}: a KL-divergence that aligns the supervisor's actual halting behaviour with the depths proposed by the planners,

$$L_{target\_depth} = \frac{1}{|S|\,K} \sum_{s \in S} \sum_{k=1}^{K} \Big(P_k^{\text{actual}} \,\big\|\, P_k^{\text{target},s} \Big), \tag{2}$$

where P_k^{actual} is the discrete probability distribution over layers obtained from the sequence of halting scores h_k^l, representing the probability that token k halts exactly at layer l. $P_k^{\text{target},s}$ is a discrete Gaussian target distribution over $l \in \{1, \ldots, L\}$:

$$P_k^{\text{target},s}(l) \propto \exp\Big[-\tfrac{(l - N^s_{\text{target},k})^2}{2\sigma^2_{\text{target}}} \Big] \tag{3}$$

with mean $N^s_{\text{target},k}$ (predicted by planner s for token k) and fixed standard deviation $\sigma_{\text{target}} = 1$.

3.5 Implementation Details

AHT-ViT is implemented in PyTorch [12]. Each planner is a two-layer MLP with ReLU activations, inserted after layers 3, 6, and 9 of the DeiT-S backbone ($L = 12$); together they add fewer than 0.5M parameters. The supervisor taps the feature vector after the first linear sub-layer of every FFN block, using a fixed halting threshold $T = 1.0$ and target-distribution width $\sigma_{\text{target}} = 1.0$. We fine-tune from the public DeiT-S checkpoint [2] for 50 epochs on ImageNet with AdamW [13] (initial learning rate 1×10^{-4}, cosine decay, weight decay 0.05) and standard data augmentations (e.g., similar to those in [2], including RandAugment [21]). The loss weights (λ_p, λ_t) are chosen by a small grid search on a held-out validation split to realise the trade-off points reported in Table 2. Conceptual pseudocode appears in Appendix A.

Table 2. Hyperparameters for AHT-ViT Configurations.

Configuration	λ_p	λ_t
High Accuracy	1×10^{-3}	1.0
Balanced	5×10^{-2}	0.5
High Efficiency	1×10^{-2}	0.2

Table 3. Performance comparison on ImageNet (DeiT-S backbone).

Model	Top-1 Acc (%)	Avg GFLOPs	Throughput (im/s)†
DeiT-S (Baseline)	79.9	4.6	1570
Evaluated Adaptive Methods:			
DynamicViT (ρ=0.7) [5]	79.3 (-0.6)	2.9 (-37%)	2590
A-ViT-S [11]	78.6 (-1.3)	3.6 (-22%)	2096
Our Results:			
AHT-ViT (High Acc Config)	79.8 (-0.1)	3.8 (-17%)	1940
AHT-ViT (Balanced Config)	79.4 (-0.5)	3.1 (-33%)	2430
AHT-ViT (High Eff Config)	78.5 (-1.4)	2.4 (-48%)	3159

†Throughput measured on NVIDIA RTX 4090 GPU with batch size 32.

4 Experiments

4.1 Setup

Datasets: ImageNet-1K [10], Places365 [15], CIFAR-100 [16]. Base Model: DeiT-S [2]. Metrics: Top-1 Accuracy, Average GFLOPs per image (calculated based on dynamic execution), Throughput (images/second, measured on NVIDIA RTX 4090 GPU, batch size 32).

4.2 Comparative Methods

We compare AHT-ViT against:

- **Static Baseline:** DeiT-S [2].
- **DynamicViT [5]:** Token pruning ($\rho = 0.7$).
- **A-ViT [11]:** Adaptive depth with global guidance.

All methods use the same DeiT-S backbone and are trained/evaluated under identical conditions (optimizer, schedule, epochs, augmentation) for fair comparison.

5 Results and Discussion

5.1 ImageNet Performance

Figure 3 and Table 3 show ImageNet results. AHT-ViT consistently improves the accuracy-efficiency trade-off over the static DeiT-S baseline. The 'Balanced Config' reduces GFLOPs by 33% with only a 0.5% accuracy drop, yielding a 1.55x throughput increase. The 'High Acc Config' matches baseline accuracy with 17% fewer GFLOPs and 1.24x throughput.

Compared to other adaptive methods evaluated under identical settings, AHT-ViT performs competitively. It achieves higher accuracy than DynamicViT ($\rho = 0.7$) at comparable efficiency (2.9 vs 3.1 GFLOPs). Notably, the AHT-ViT 'Balanced' configuration surpasses A-ViT by +0.8% accuracy while using fewer GFLOPs (3.1 vs 3.6) and achieving higher throughput (2430 vs 2096 im/s). This suggests the explicit planning mechanism (L_{target_depth}) provides a tangible benefit over global distributional priors for guiding adaptive depth in this setup. Throughput gains generally track GFLOPs reductions but are slightly lower proportionally, likely due to the small overhead of the adaptive halting logic.

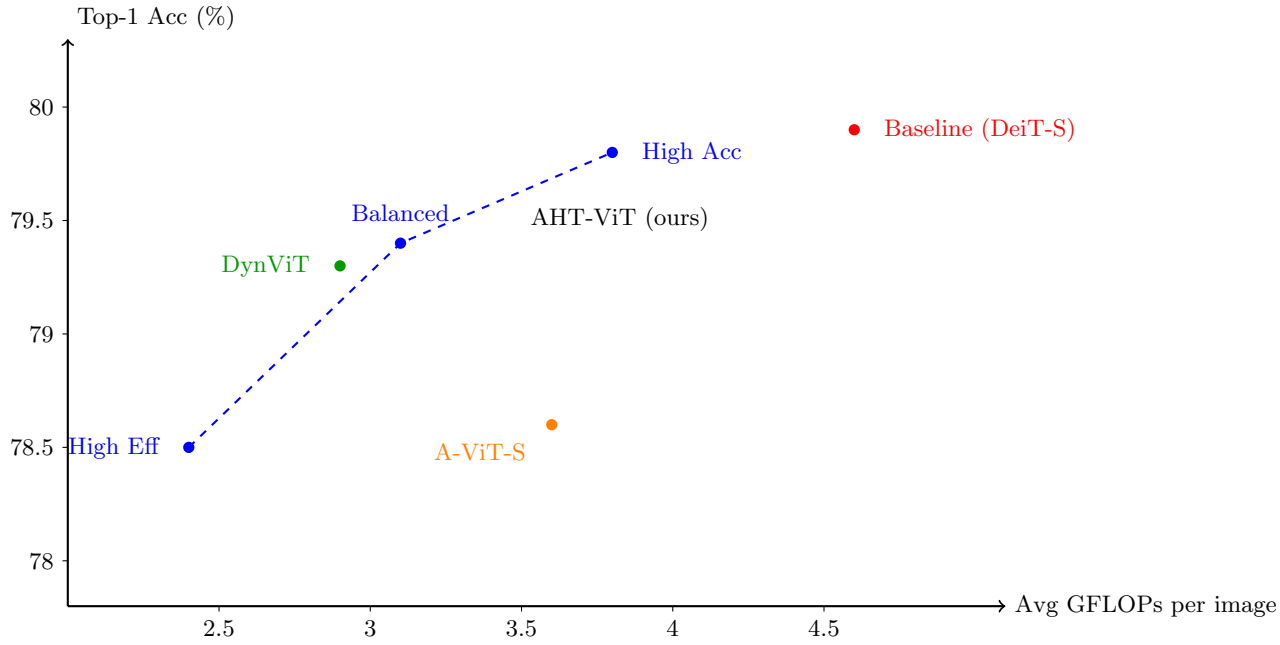


Fig. 3. Accuracy vs. GFLOPs on ImageNet. AHT-ViT (blue) compared to baseline DeiT-S (red) and our evaluations of DynamicViT (green) and A-ViT-S (orange). Points are annotated with configuration/method.

5.2 Performance on Other Datasets

The AHT-ViT 'Balanced' configuration, initially trained on ImageNet, was fine-tuned on Places365 [15] and CIFAR-100 [16]. As shown in Table 4, the model retained its efficiency advantages (e.g., reduced GFLOPs and improved throughput, comparable to those on ImageNet) while incurring only moderate accuracy drops compared to the static baseline (-1.4% on Places365, -0.5% on CIFAR-100). This demonstrates that the learned adaptive strategy generalizes reasonably well across datasets.

Table 4. AHT-ViT ('Balanced' ImageNet Config) Performance on Other Datasets vs. DeiT-S Baseline.

Dataset	Model	Accuracy (%)	Δ Acc (%)	Avg GFLOPs	Δ GFLOPs (%)	Throughput (im/s)[†]
Places365	DeiT-S	81.6	-	4.6	-	1570
	AHT-ViT	80.2	-1.4	2.8 (-39%)	-39%	2579
CIFAR-100	DeiT-S	90.1	-	4.6	-	1570
	AHT-ViT	89.6	-0.5	3.7 (-20%)	-20%	1952

[†]Throughput measured on NVIDIA RTX 4090 GPU with batch size 32.

5.3 Ablation Study: Role of Target Depth Guidance

Removing the L_{target_depth} loss ($\lambda_t = 0$) and re-tuning λ_p yielded inferior results (Table 5), demonstrating lower accuracy and throughput at comparable GFLOPs. This ablation confirms the benefit of explicitly guiding the supervisor execution using the planner predictions via the L_{target_depth} loss, compared to relying solely on the L_{ponder} cost penalty.

Table 5. Ablation study on the L_{target_depth} guidance (ImageNet).

Model Variant	Top-1 Acc (%)	Avg GFLOPs	Throughput (im/s)[†]
AHT-ViT (Full, Balanced Config)	79.4	3.1	2430
AHT-ViT (w/o L_{target_depth})	78.8	3.4	2124

[†]Throughput measured on NVIDIA RTX 4090 GPU with batch size 32.

5.4 Qualitative Halting Behavior

To provide insight into the learned adaptive behavior, Figure 1 illustrates the halting process on sample images. Each image displays the original input on the left and the result after adaptive halting on the right. Patches with the overlay on the right side represent tokens that were halted early by the supervisor mechanism, reducing computation. It can be observed that tokens corresponding to simpler or background regions (like the blurred background in the second example or uniform textures in the first) are often halted earlier, while tokens representing more complex foreground features (like the cats' faces and fur patterns) are processed deeper into the network. This demonstrates meaningful adaptation to image content. Further visualization studies across diverse images and classes are valuable future work [14].

6 Conclusion

We presented AHT-ViT, an adaptive halting transformer enhancing ViT efficiency via a planner-supervisor architecture. Hierarchical planners predict target depths, while a highly efficient supervisor executes halting, guided by a novel KL divergence-based loss (L_{target_depth}) aligning execution with plans. Experiments on ImageNet, Places365, and CIFAR-100 demonstrated improved accuracy-FLOPs/throughput trade-offs compared to static baselines and competitive performance against directly evaluated adaptive methods (DynamicViT, A-ViT). The explicit planning mechanism appears beneficial over global guidance or token pruning in our setup.

Key limitations include the need for evaluation on broader tasks (detection, segmentation) and larger models. Further ablation studies on planner configurations, halting threshold T, and target variance σ_{target} are warranted. While throughput was measured, exploring latency on diverse hardware remains important. AHT-ViT provides a promising approach for efficient ViTs through planned, dynamic computation depth.

References

1. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.
2. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H. (2021, July). Training data-efficient image transformers distillation through attention. In International conference on machine learning (pp. 10347-10357). PMLR.
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
4. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y. (2021). Dynamic neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence, 44(11), 7436-7456.
5. Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., Hsieh, C. J. (2021). Dynamicvit: Efficient vision transformers with dynamic token sparsification. Advances in neural information processing systems, 34, 13937-13949.
6. Kong, Z., Dong, P., Ma, X., Meng, X., Sun, M., Niu, W., ... Wang, Y. (2022). SPViT: enabling faster vision transformers via soft token pruning. URL https://arxiv. org/abs/2112.13890.

7. Graves, A. (2016). Adaptive computation time for recurrent neural networks. arXiv preprint arXiv:1603.08983.
8. Figurnov, M., Collins, M. D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., Salakhutdinov, R. (2017). Spatially adaptive computation time for residual networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1039-1048).
9. Kullback, S., Leibler, R. A. (1951). On information and sufficiency. The annals of mathematical statistics, 22(1), 79-86.
10. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.
11. Yin, H., Vahdat, A., Alvarez, J. M., Mallya, A., Kautz, J., Molchanov, P. (2022). A-vit: Adaptive tokens for efficient vision transformer. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 10809-10818).
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32.
13. Loshchilov, I., Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.
14. Chefer, H., Gur, S., Wolf, L. (2021). Transformer interpretability beyond attention visualization. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 782-791).
15. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A. (2017). Places: A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence, 40(6), 1452-1464.
16. Krizhevsky, A., Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical Report, University of Toronto.
17. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 10012-10022).
18. Teerapittayanon, S., McDanel, B., Kung, H. T. (2016, December). Branchynet: Fast inference via early exiting from deep neural networks. In 2016 23rd international conference on pattern recognition (ICPR) (pp. 2464-2469). IEEE.
19. Bolya, D., Fu, C. Y., Dai, X., Zhang, P., Feichtenhofer, C., Hoffman, J. (2022). Token merging: Your vit but faster. arXiv preprint arXiv:2210.09461.
20. Banino, A., Balaguer, J., Blundell, C. (2021). Pondernet: Learning to ponder. arXiv preprint arXiv:2107.05407.
21. Cubuk, E. D., Zoph, B., Shlens, J., Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops (pp. 702-703).
22. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S. (2020, August). End-to-end object detection with transformers. In European conference on computer vision (pp. 213-229). Cham: Springer International Publishing.
23. Thisanke, H., Deshan, C., Chamith, K., Seneviratne, S., Vidanaarachchi, R., Herath, D. (2023). Semantic segmentation using Vision Transformers: A survey. Engineering Applications of Artificial Intelligence, 126, 106669.
24. Menghani, G. (2023). Efficient deep learning: A survey on making deep learning models smaller, faster, and better. ACM Computing Surveys, 55(12), 1-37.
25. Liang, Y., Ge, C., Tong, Z., Song, Y., Wang, J., Xie, P. (2022). Not all patches are what you need: Expediting vision transformers via token reorganizations. arXiv preprint arXiv:2202.07800.
26. Tang, Y., Han, K., Wang, Y., Xu, C., Guo, J., Xu, C., Tao, D. (2022). Patch slimming for efficient vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 12165-12174).
27. Wang, S., Li, B. Z., Khabsa, M., Fang, H., Ma, H. (2020). Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768.
28. Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., ... Weller, A. (2020). Rethinking attention with performers. arXiv preprint arXiv:2009.14794.
29. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
30. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

31. Wang, W., Xie, E., Li, X., Fan, D. P., Song, K., Liang, D., ... Shao, L. (2021). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 568-578).
32. Chen, C. F. R., Fan, Q., Panda, R. (2021). Crossvit: Cross-attention multi-scale vision transformer for image classification. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 357-366).
33. Liu, W., Zhou, P., Zhao, Z., Wang, Z., Deng, H., Ju, Q. (2020). Fastbert: a self-distilling bert with adaptive inference time. arXiv preprint arXiv:2004.02178.
34. Xin, J., Tang, R., Lee, J., Yu, Y., Lin, J. (2020). DeeBERT: Dynamic early exiting for accelerating BERT inference. arXiv preprint arXiv:2004.12993.
35. Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., ... Chen, Z. (2020). Gshard: Scaling giant models with conditional computation and automatic sharding. arXiv preprint arXiv:2006.16668.
36. Fedus, W., Zoph, B., Shazeer, N. (2022). Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Journal of Machine Learning Research, 23(120), 1-39.
37. Kingma, D. P., Welling, M. (2013, December). Auto-encoding variational bayes.

A Conceptual Code

Listing 1.1 shows the conceptual core loop logic referenced in Section 3.5.

Listing 1.1. Conceptual AHT-ViT Core Loop Logic (Appendix).

```
import torch
import torch.nn as nn
# Assume: self.layers, self.planners, self.planner_stages
# Assume: self.gamma, self.beta (shared nn.Parameters)
# Assume: self.THRESHOLD defined

def forward(self, x_tokens): # Inside AHT_ViT class
    B, N, _ = x_tokens.shape
    cum_scores = torch.zeros(B, N, device=x_tokens.device)
    active_mask = torch.ones(B, N, dtype=torch.bool, ...)
    plan_targets = {} # Store planner outputs

    current_tokens = x_tokens
    for l in range(1, self.L + 1):
        # 1. Process layer (returns output & supervisor features)
        # Layer must handle masking or expect masked input
        out_tokens, supervisor_features = self.layers[l-1](
            current_tokens, attention_mask=active_mask ) # Pass mask
        # 2. Supervisor: Calc halting score h_l for active tokens
        # supervisor_features assumed shape [B, N, FeatureDim]
        selected_feature = supervisor_features[..., 0] # Use first dim
        h_scores_l = torch.sigmoid(
            self.gamma * selected_feature + self.beta
            ) * active_mask.float() # Score is 0 if already halted
        # 3. Accumulate score & update active_mask for next layer
        cum_scores += h_scores_l
        active_mask_next = (cum_scores < self.THRESHOLD)
        # 4. Planner: Predict target if stage
        if l in self.planner_stages:
            # Planner might need masking internally too
            plan_targets[l] = self.planners[str(l)](out_tokens) * \
                              active_mask.float() # Mask output for
                                  halted tokens

        active_mask = active_mask_next # Update mask for next iteration
        current_tokens = out_tokens
```