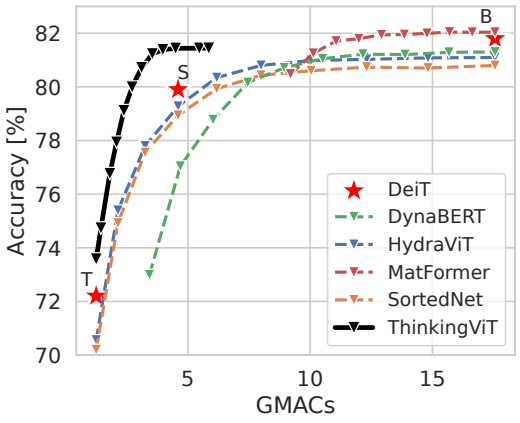
arXiv:2507.10800v2 [cs.CV] 17 Nov 2025

ThinkingViT: Matryoshka Thinking Vision Transformer for Elastic Inference

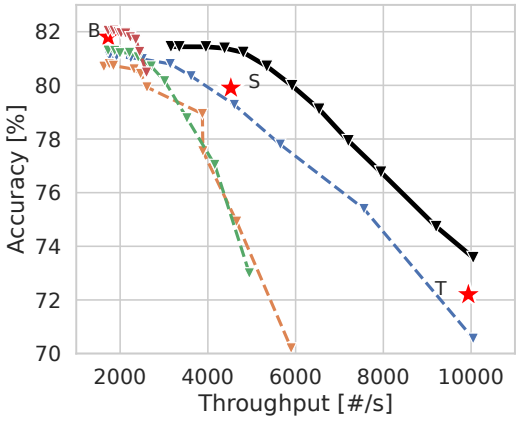
Ali Hojjat[1,2], Janek Haberer[1], Sören Pirk[1], Olaf Landsiedel[2,1]
[1] Kiel University, Germany [2] Hamburg University of Technology (TUHH), Germany
{ali.hojjat, olaf.landsiedel}@tuhh.de {janek.haberer, soeren.pirk}@cs.uni-kiel.de
github.com/ds-kiel/ThinkingViT

Abstract

Vision Transformers (ViTs) deliver SOTA performance, yet their fixed computational budget prevents scalable deployment across heterogeneous hardware. Recent Matryoshka-style Transformer architectures mitigate this by embedding nested subnetworks within a single model to enable scalable inference. However, these models allocate the same amount of compute to all inputs, regardless of their complexity, which leads to inefficiencies. To address this, we introduce ThinkingViT, a nested ViT architecture that employs progressive thinking stages to dynamically adjust inference computation based on input difficulty. ThinkingViT first activates a small subset of the most important attention heads to produce an initial prediction. If the prediction confidence exceeds a predefined threshold, inference terminates early. Otherwise, within the same backbone, it activates a larger subset of attention heads and conducts a new forward pass. This process continues iteratively until the model reaches the predefined confidence level or exhausts its maximum capacity. To boost the performance of subsequent rounds, we introduce a Token Recycling approach that fuses the input embeddings with the embeddings from the previous stage. Experiments show that ThinkingViT surpasses nested baselines by up to 2.0 percentage points (p.p.) in accuracy at the same throughput and by up to 2.9 p.p. at equal GMACs on ImageNet-1K. We show that the backbone-preserving design of ThinkingViT allows it to serve as a plug-in upgrade for ViTs in downstream tasks such as semantic segmentation. We also demonstrate that ThinkingViT transfers effectively to other architectures such as Swin.



(a) GMACs vs. Accuracy on ImageNet-1K



(b) Throughput vs. Accuracy on ImageNet-1K

Figure 1. Comparison of *ThinkingViT* with MatFormer(NeurIPS'24) [8], HydraViT(NeurIPS'24) [14], SortedNet(NeurIPS'23-W) [43], and DynaBERT(NeurIPS'20) [19], evaluated in terms of GMACs and throughput on an A100. All baselines have a dynamic width within a standard ViT backbone and are trained following the training recipes in [41]. *ThinkingViT* is trained with two progressive thinking stages using 3 and 6 heads, and consistently surpasses baseline by up to 2.0 p.p. at the same throughput and by up to 2.9 p.p. at the same GMACs. See Appendix H for a comparison with smaller baseline models.

1. Introduction

Motivation: Vision Transformer (ViT) [10] have achieved state-of-the-art results across numerous image recognition tasks [15, 28], yet their non-elastic inference pipelines impose a uniform and often excessive computational cost. This lack of flexibility poses a significant limitation in real-world deployments, where devices vary widely in computational

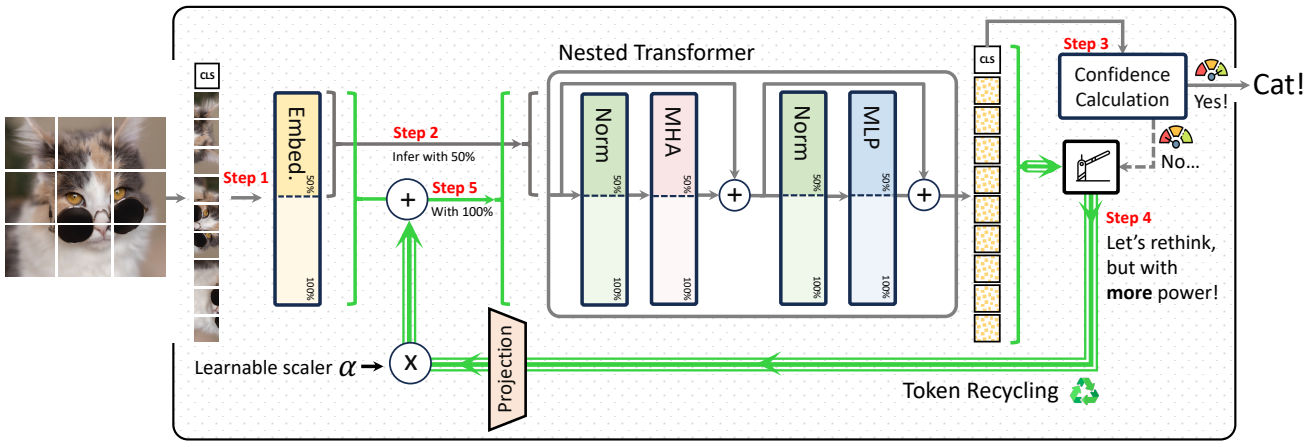


Figure 2. **Nested progressive inference with Token Recycling in *ThinkingViT*.** After embedding the input, *ThinkingViT* first activates a subset of the model, including the first attention heads (e.g., 50%), to produce an initial prediction. Due to the training procedure, these heads capture the most important features. If the certainty exceeds a threshold (easy inputs), inference terminates early to save computation. Otherwise, the resulting tokens are fused back into the input via a learnable scaling factor α, which controls how much prior knowledge is recycled. The model then *thinks more* by reprocessing the fused tokens using a larger subset of the attention heads (e.g., 100%) for a refined prediction. *ThinkingViT* enables elastic inference across different hardware budgets by adjusting the confidence threshold. The number of thinking stages and the proportion of attention heads activated at each stage can be flexibly configured based on efficiency and accuracy trade-offs, see Section 4.2.

power and latency constraints [49]. From high-throughput servers to mobile and embedded platforms, modern applications require elastic inference [4], where a model can adjust its computational footprint to match the capabilities of the target hardware. Without such adaptability, ViTs struggle to meet the performance and efficiency tradeoffs necessary for scalable and widespread deployment.

Limitations of Prior Work: Recent progress in *nested* Transformer architectures offers a compelling pathway toward elastic inference [8, 14, 54]. These methods embed multiple nested subnetworks within a single Transformer backbone, enabling dynamic subnetwork selection at inference time. This strategy allows models to operate under varying latency and hardware constraints without any extra tuning or post-training adjustments, all while maintaining a unified parameter set. However, these approaches typically allocate a *fixed* computational budget per input, missing the opportunity to distinguish between simple and complex samples, which leads to inefficiencies, especially under tight resource constraints.

Image-Based Routing: To make nested models adaptable, we must route the input image to the appropriate subnetwork based on its difficulty. While the concept of routing appears in Mixture of Experts (MoE) models [58], we cannot use them out of the box since these routers operate at the token level by directing token embeddings via lightweight Multilayer Perceptrons (MLPs), which struggle to capture the global complexity of the image. In practice, accurately estimating image difficulty demands representational power comparable to that of a full-scale classifier, a requirement that such compact routers cannot meet [9, 31]. Moreover, dedicating substantial compute to a separate router that merely forwards the input to another model introduces overhead. This leads to a critical design question: *How can we enable input-aware compute allocation within a **nested** Vision Transformer, without relying on a separate costly routing mechanism?*

***ThinkingViT*:** Inspired by recent advances in thinking-based Large Language Modelss (LLMs) [13, 38], we introduce *ThinkingViT*, a ViT built upon the vanilla ViT architecture [10] that dynamically adjusts its inference effort based on input complexity through *nested progressive thinking stages*. *ThinkingViT* begins by activating a subset of the nested network, including the most important attention heads (e.g., the top 50%) to generate an initial prediction along with a certainty score. If the certainty is sufficiently high (the "aha" moment [13]), inference terminates early. Otherwise, the processed embeddings are fused with the original input embeddings and passed through a larger subset of the nested network, using an increased set of attention heads (e.g., 100%) to perform a more thorough re-evaluation. This token fusion mechanism allows the model to avoid thinking from scratch; instead, it **recycles** the knowledge gained in the first pass, refining its prediction with improved accuracy; see Figure 2. Our proposed progressive thinking mechanism enables the model not only to *think more*, but also to *think more powerfully* when processing ambiguous inputs, ensuring that more challenging examples receive greater representational capacity, i.e., more attention heads. In this pipeline, the prediction certainty serves as the central mecha-

Table 1. Looping over the same model offers limited accuracy gains despite increased computation.

Model	Depth	Size [M]	GMACs	Acc.
DeiT-Tiny	12	5.70	1.25	72.20
Naïve Iteration ↻				
+ 1× iteration	24	5.70	2.5	74.00
+ 2× iteration	36	5.70	3.75	74.10
+ 3× iteration	48	5.70	5.00	73.60

nism that governs elastic inference: the model halts early for cases with high certainty and progressively deepens computation for more uncertain inputs. By adjusting the certainty threshold, the model naturally balances performance against efficiency, supporting elastic inference without requiring any separate routing mechanism. Additionally, nesting all stages within a single unified model avoids parameter duplication and enables more efficient training and improved accuracy.

Furthermore, unlike iterative refinement in LLMs that repeatedly use the same model [13], naïve iterative chains that simply refeed a ViT's own outputs into the same network fail to deliver consistent improvements and quickly saturate in vision models; see Table 1. These findings underscore the need for the progressive expansion strategy adopted in *ThinkingViT*.

Contributions:

1. We introduce *ThinkingViT*, a thinking-based Vision Transformer (ViT) that brings input adaptivity to nested Transformers by dynamically adapting computation based on image difficulty.
2. *ThinkingViT* executes multiple rounds of inference by progressively activating larger subsets of attention heads, allowing the model to halt early for easy inputs based on the certainty of the predictions, while allocating greater capacity to harder examples that require richer representations.
3. *ThinkingViT* introduces Token Recycling to condition each subsequent round of inference on the features produced in the previous round, improving overall accuracy.
4. *ThinkingViT* achieves up to 2.0 percentage points higher accuracy at equal throughput, and up to 2.9 points higher at the same GMACs compared to baseline models (see Figure 1), and extends to hierarchical architectures and downstream tasks, confirming broad adaptability.

2. Related Work

Nested Models: Slimmable networks first demonstrated that a single model can operate at multiple widths using shared weights and width-specific normalization [51, 52]. Many subsequent works build on this idea. For instance, MatFormer [8], based on Matryoshka Representation Learning [26], introduces multiple nested subsets within the hidden layer of MLP. DynaBERT [19] slices the Multi-head Attention (MHA) and MLP layers, but does not slice embeddings across layers since it relies on knowledge distillation. SortedNet [43] generalizes nesting across MLPs, Normalization Layer (NORM), MHA, and embedding dimensions, although it retains a fixed number of attention heads. HydraViT [14] and Slicing ViT [54] enable slicing across embeddings, NORM, MLP, and MHA, and support a dynamic number of heads. However, like the other mentioned methods, they apply a fixed compute budget per input, limiting their ability to adapt based on input complexity and leading to inefficiencies under constrained resources.

Routing: Routing mechanisms got popular in the MoE framework [58] and have since been extended to nested designs such as MoNE [21], MoD [34], and AMoD [12]. Flextron introduces a surrogate loss predictor to guide token routing [4]. These methods typically rely on lightweight MLPs, which lack the representational capacity to route at the image level, where decisions often demand the reasoning power of a full classifier. Ensemble-based strategies like Selective Query [23], OCCAM [9], and RouteLLM [31] perform input-aware routing using full models as gates. However, these approaches operate over a fixed set of pretrained models (e.g., CNNs or Transformers) and do not enable any knowledge transfer between the router and the routed models. In contrast, *ThinkingViT* recycles tokens across stages, allowing each round to build on previous inferences for improved prediction.

Thinking: Recent advancements in LLM have introduced mechanisms for adaptive reasoning depth [11, 32]. These methods often use reinforcement learning to dynamically adjust the number of reasoning steps based on input complexity [13, 38]. However, excessive reasoning length can inflate inference cost without proportional gains in accuracy [25]. Motivated by this, recent efforts seek to preserve the benefits of deep reasoning while avoiding redundant computation by introducing test-time adaptability or confidence-based early stopping [30, 53]. These models, however, reuse the same network in each round, which, as reported in Table 1, results in limited gains. In contrast, *ThinkingViT* scales up its capacity during rethinking to overcome this challenge.

Early-exit: Early-exit methods reduce inference cost by attaching intermediate classifiers that let models stop once predictions are confident. Classic approaches such as BranchyNet [40], MSDNet [20], and SDN [24] introduced multi-exit architectures, later extended to Transformers with PABEE [57] and BERxiT [47], and to ViTs with ViT-EE [1], PCEE [55], and LGViT [48]. Jointly learned strategies such as JEI-DNN [36] further improve reliability by co-training gating and exit classifiers. While these methods offer dynamic depth, they rely on fixed-width models; in contrast, *ThinkingViT* performs multiple loops over the model with progressively increased width.

3. *ThinkingViT*

How *ThinkingViT* works: After embedding the input, *ThinkingViT* runs inference with a small subset of attention heads that, due to the training procedure, are the most important heads. The model stops if the classification confidence is high. Otherwise, within the same backbone, it activates a larger subset of attention heads and performs another inference step. This iterative expansion continues until the model reaches the predefined confidence or exhausts its maximum capacity. Token Recycling further improves later stages by fusing input embeddings with features from previous stages. By adjusting the confidence threshold, *ThinkingViT* enables elastic inference across different target latencies.

3.1. Nested Vision Transformer - Sorting Heads

ThinkingViT is built on top of the standard ViT architecture. Let $V_{D,H}$ be a ViT model with H attention heads and the embedding dimension of D. ViT begins by tokenizing the input image x into P non-overlapping patches, each of which is projected into a D-dimensional embedding vector $\mathcal{E}^D$ using a CNN-based patch embedding function. After adding positional encodings, the resulting sequence is processed by L standard Transformer blocks:

$$z^l = \text{Block}^l(z^{l-1}) \quad \text{for } l = 1, \ldots, L \tag{1}$$

Inducing sorted subnetworks: Inspired by [14], *ThinkingViT* induces n sorted subnetworks within this architecture. Each subnetwork is denoted as V_{d_i,h_i} and is constructed using the first d_i embedding values of each token and the first h_i attention heads from the full model. To build such a nested structure, *ThinkingViT* slices all components of the ViT, including the embedding layer, attention modules, MLP blocks, and normalization layers, according to these indices. This yields contained subnetworks with progressively increasing embedding dimensionality and attention capacity:

$$V_{d_1,h_1}(x) \subset V_{d_2,h_2}(x) \subset \cdots \subset V_{d_n,h_n}(x), \tag{2}$$

$$d_1 < d_2 < \cdots < d_n, \quad h_1 < h_2 < \cdots < h_n \tag{3}$$

3.2. Looping nested Vision Transformers through Token Recycling

Inspired by recent advances in reasoning models [13, 38], *ThinkingViT* operates through multiple rounds of progressive refinement using our proposed "Token Recycling" mechanism. Although in practice, we find that two subnetworks are sufficient to achieve strong performance (as demonstrated in Figure 3), we present the general multi-round formulation here for clarity and broader applicability.

After constructing the nested architecture, *ThinkingViT* begins by using the smallest subnetwork V_{d_i,h_i}, with the embedding dimension of d_i and h_i attention heads. This subnetwork processes the input image to predict the class and generate token embeddings z^L. Subsequently, to refine its prediction, the model expands computational capacity by activating a larger subnetwork $V_{d_j,h_j}(x)$, featuring h_j attention heads, where $h_i < h_j$ and $d_i < d_j$. Afterwards, it fuses the produced token embeddings z^L with the new input embeddings $\mathcal{E}^{d_j}(x) \in \mathbb{R}^{P \times d_j}$, through a projection layer and scaled by a learnable parameter α, determining the weight of embeddings "recalled" from the previous step:

$$\mathcal{E}^{d_j}_{\text{fused}} = \alpha \cdot \text{Proj}_{d_i \to d_j}(z^L) + \mathcal{E}^{d_j}(x), \tag{4}$$

This *Token Recycling* mechanism allows the model to reuse prior representations instead of reprocessing from scratch. The process continues iteratively and, unlike language models that reuse the same network for reasoning [13], our approach increases model capacity by activating progressively larger subsets of attention heads, enhancing representational capacity at each step. Progressive expansion is essential in vision tasks, where naive recursion quickly plateaus, as shown in Table 1. Appendix E and Appendix D compare recycling and fusion strategies in detail.

3.3. Training all subnetworks jointly

During training, *ThinkingViT* executes all n thinking rounds and minimizes a weighted classification loss $\mathcal{L}_{\text{cls}}$ across all stages:

$$\mathcal{L} = \sum_{i=1}^{n} \lambda_i \cdot \mathcal{L}_{\text{cls}}(V_{d_i,h_i}(x), y) \tag{5}$$

where y is the ground-truth label and λ_i controls the contribution of each subnetwork to the global objective. For a small number of subnetworks, it is computationally feasible to optimize all subnetworks simultaneously, as the gradient computation graph remains reasonably compact. However, as n increases, training all subnetworks jointly becomes computationally intensive. In such cases, we adopt strategies such as the sandwich rule [51] and stochastic subnetwork sampling [14] to reduce overhead.

3.4. The "Aha!" moment

By inducing nested iterative subnetworks inside the model, *ThinkingViT* builds a hierarchy of n nested thinking steps, where each step operates on top of the previous one to refine the prediction. However, different inputs have different complexities, and for some inputs, the model becomes confident after only k iterations, where $k < n$ (e.g., after a single round of thinking). The "aha!" moment occurs when the model recognizes that it has reached sufficient certainty, allowing it to allocate an appropriate number of thinking rounds based on the input complexity [13]. Let f_k denote the softmax output produced at the k-th iteration. After the k^{th} round, the model measures its certainty using Shannon

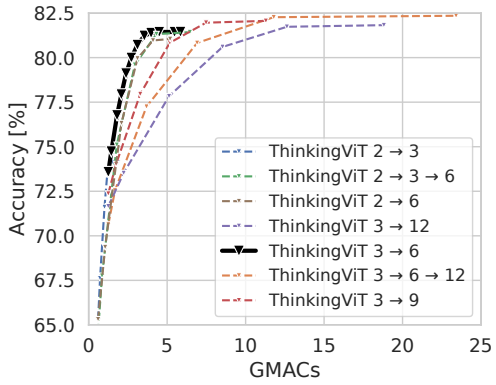


Figure 3. Accuracy vs. GMACs for *ThinkingViT* variants on ImageNet-1K. 3H → 6H achieves the best trade-off, while 2H → 3H → 6H covers the widest range with only a slight accuracy drop. See Appendix K for details.

entropy:

$$\mathcal{H}(f_k) = -\sum_{c=1}^{C} f_k^{(c)} \log f_k^{(c)}, \tag{6}$$

where C is the number of classes. If the entropy $\mathcal{H}(f_k)$ is lower than a predefined threshold τ, inference halts early and the model accepts the current prediction. Otherwise, *ThinkingViT* activates a larger subnetwork to further refine the result. This simple metric performs on par with more complex criteria [22] on ImageNet-1K. Our design also allows alternative routing modules [9, 23] to be integrated as drop-in replacements without changing the architecture or training.

3.5. Elastic inference

ThinkingViT supports elastic inference by using its own certainty to determine when to stop. The entropy threshold τ governs the trade-off between efficiency and accuracy. A low threshold enforces stricter confidence requirements, causing most inputs to undergo more thinking steps, which increases computational cost but improves accuracy. In contrast, a high threshold allows earlier exits for more inputs, reducing latency and computation cost at the expense of potential accuracy loss. This flexibility enables users to tune the model at runtime according to specific resource and performance needs.

4. Evaluation

In this section, we analyze how entropy guides *ThinkingViT*'s adaptive computation (Sec. 4.2), evaluate it on ImageNet-1K and its variants against SOTA nested baselines (Sec. 4.3), extend the evaluation to hierarchical such as Swin Transformer (Sec. 4.6) and segmentation architectures (Sec. 4.5), compare with early-exit baselines (Sec. 4.7), and finally assess robustness on uniformly challenging datasets (Sec. 4.8).

4.1. Setup

Implementation details: We train *ThinkingViT* on ImageNet-1K [37] at a resolution of 224×224, using models implemented in `timm` [46] and following the setup of Touvron et al. [41], with all models initialized with a pretrained DeiT-Tiny checkpoint. We train on a cluster with NVIDIA H100 GPUs, taking about 10 minutes per epoch on 2 GPUs. During prototyping, we conducted roughly 100 training runs, each lasting 300 epochs.

Baselines: We compare *ThinkingViT* against several width expansion baselines: MatFormer [8], which slices only the hidden layer of MLP while keeping MHA and embedding dimensions fixed; DynaBERT [19], which slices both MHA and the hidden layer of MLP while keeping embeddings fixed; SortedNet [43], which slices all embeddings, NORM, MHA, and MLP, while keeping the number of heads fixed; and HydraViT [14], which slices MHA, embeddings, NORM, and MLP, while also changing the number of heads. Similar to HydraViT, *ThinkingViT* adopts a slicing strategy across the number of heads, MHA, embeddings, NORM, and MLP layers. However, unlike the above methods that follow static inference paths regardless of input difficulty, *ThinkingViT* introduces input-adaptive computation. In Section 4.7, we present comparisons to depth-expansion baselines.

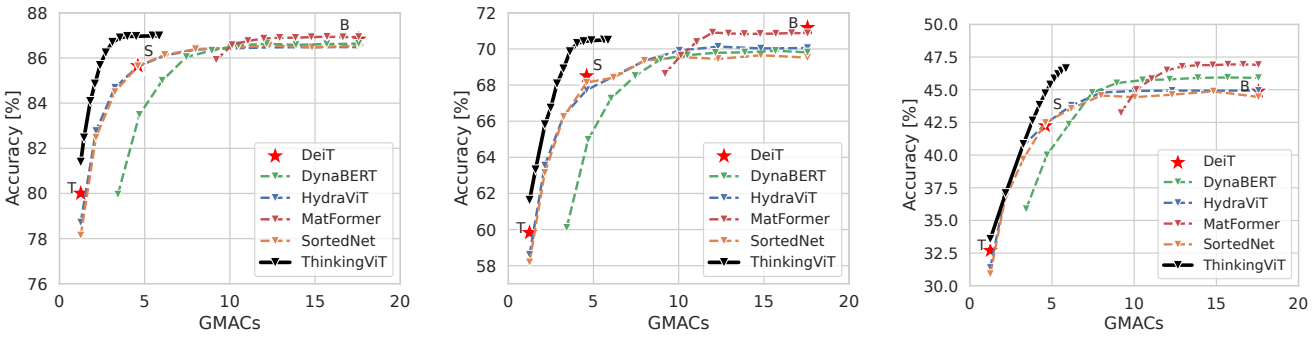
Throughput and GMACs: We measure throughput and computational cost on a single NVIDIA A100 GPU with a batch size of 512. To evaluate throughput, we run the model under different confidence thresholds τ. For each threshold, we record the per-image inference time across the entire validation set and compute the average throughput as the total number of images processed divided by the total inference time. By varying the threshold τ, we obtain the throughput curves reported in the plots. For further details, see Appendix J. For GMACs, we follow a similar procedure. The total compute per image for threshold τ is then averaged over all samples as

$$\text{GMACs}_\tau = \sum_{i=1}^{N} p_i \cdot \left(\sum_{j=1}^{i} \text{GMACs}_j \right),$$

where p_i denotes the fraction of samples that stop at round i, and GMACs_j denotes the GMACs of round j. By varying the threshold τ, we obtain the GMACs curves reported in the plots.

4.2. Trade-offs in attention expansion

ThinkingViT expands computational capacity through progressive activation of attention heads. Generally, the number of thinking stages and the attention head expansion



(a) ImageNet-Real (b) ImageNet-V2 (c) ImageNet-R

Figure 4. GMACs vs. Accuracy on ImageNet variants. *ThinkingViT* has superior performance compared to the baselines.

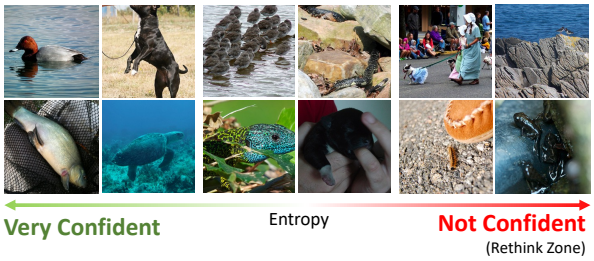


Figure 5. Visualization of images sorted by first-round entropy. *ThinkingViT* confidently classifies simple, clear images in one round, while complex cases with occlusion or clutter show higher entropy and trigger a second round.

step size are hyperparameters, and their optimal configuration depends on the dataset and target efficiency objectives. In Figure 3, we evaluate this trade-off and find that the 3H $\rightarrow$ 6H configuration offers the most favorable balance between accuracy and compute on ImageNet-1K. The 2H $\rightarrow$ 3H $\rightarrow$ 6H variant spans the widest GMAC range among the high-performing models, while incurring only a small drop in accuracy compared to 3H $\rightarrow$ 6H. At the upper end, 3H $\rightarrow$ 6H $\rightarrow$ 12H achieves the highest final accuracy, surpassing 3H $\rightarrow$ 6H by 0.91 percentage points (p.p.), but raises the compute cost from 5.85 GMACs to 23.41 GMACs for only a marginal gain. This substantial increase in cost leads to lower overall efficiency compared to the 3H $\rightarrow$ 6H setup, underscoring the importance of aligning *ThinkingViT*'s thinking strategy with specific deployment goals. Since we focus on configurations that offer the best trade-off between accuracy and GMACs, we adopt 3H $\rightarrow$ 6H for subsequent experiments. For additional experiments and a larger depiction of Figure 3, see Appendix K.

4.3. Comparing *ThinkingViT* and baselines

Figure 1 compares *ThinkingViT* with SOTA baselines in terms of GMACs and inference throughput, using the 3H $\rightarrow$ 6H configuration. *ThinkingViT* achieves up to 2.0 p.p. higher accuracy at equal throughput, and up to 2.9 p.p. higher accuracy at the same GMACs compared with its baselines. This improvement stems from three key factors. *First*, *ThinkingViT* makes early predictions using only 3 heads for easy inputs and expands to 6 heads only when needed. *Second*, *ThinkingViT* fuses embeddings from the first stage into the second, leading to better performance than flat 6-head models like HydraViT or DeiT-S. Moreover, *ThinkingViT* achieves 81.44% with only 22.01 M params, which is just 0.36 p.p. lower than DeiT-Base with 86.6 M params. This result further validates the effectiveness of the Token Recycling strategy. Importantly, it also demonstrates that achieving higher accuracy does not necessarily require a larger model; rather, it depends on applying sufficient computation. *Third*, baselines typically employ multiple slicing points throughout the architecture to meet diverse GMACs. However, optimizing for all such configurations can lead to accuracy reduction [14]. *ThinkingViT* sidesteps this issue by supporting elastic inference using just a few nested subnetworks, with compute budget adaptability controlled via the entropy threshold. We report detailed performance results in the Appendix A.

To assess robustness, we evaluate *ThinkingViT* on ImageNet-V2 [35], ImageNet-ReaL [2], and on ImageNet-R [18] in Figure 4. Additional results on ImageNet-A [17] and ImageNet-Sketch [45] are provided in Appendix B. Across all robustness benchmarks, *ThinkingViT* consistently surpasses the baselines, demonstrating superior generalization under distribution shifts. Furthermore, on ImageNet-ReaL, -Sketch, and -R, *ThinkingViT* exceeds the accuracy of DeiT-Base, despite using significantly fewer parameters (22.1M vs. 86.6M) and lower GMACs (5.85 vs. 17.56), which highlights the effectiveness of the *Token Recycling*. We further compare and combine *ThinkingViT* with "token-based pruning" approaches in Appendix G, showing that *ThinkingViT* both outperforms these methods and remains complementary with them.

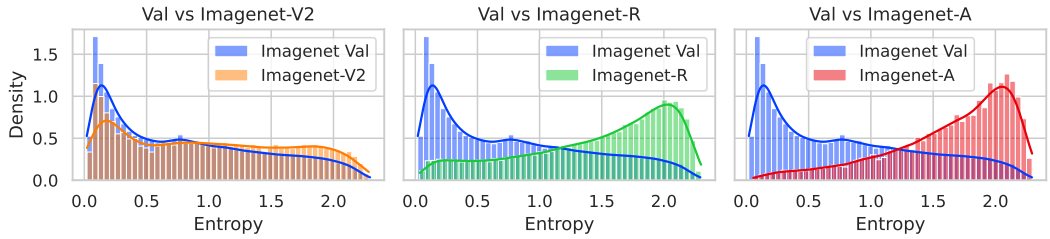


Figure 6. Entropy distribution after the first inference round across ImageNet validation sets. Simpler datasets like ImageNet-V2 show confident early predictions (left-skewed), while harder ones like ImageNet-A and -R show greater uncertainty (right-skewed), which triggers *ThinkingViT* to go for the next round of thinking.

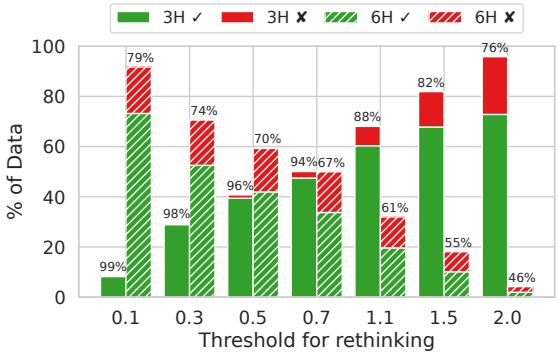
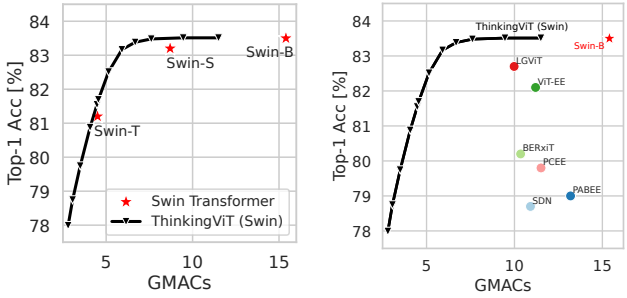


Figure 7. Load distribution of *ThinkingViT* 3H → 6H across entropy thresholds. Bars indicate usage share; numbers show accuracy (green: correct, red: incorrect) on ImageNet-1K. High 3H accuracy at low entropy confirms the effectiveness of entropy-based routing.



(a) Swin Transformer (b) Early-Exit

Figure 8. (a): *ThinkingViT* with a Swin backbone outperforms standard Swin models, matching Swin-Base accuracy with far fewer GMACs. (b): Comparison of *ThinkingViT* with Early-Exit baselines (with Swin backbones), showing that iterative refinement provides improvement beyond early termination strategies.

4.4. Analyzing entropy as a signal to think deeper

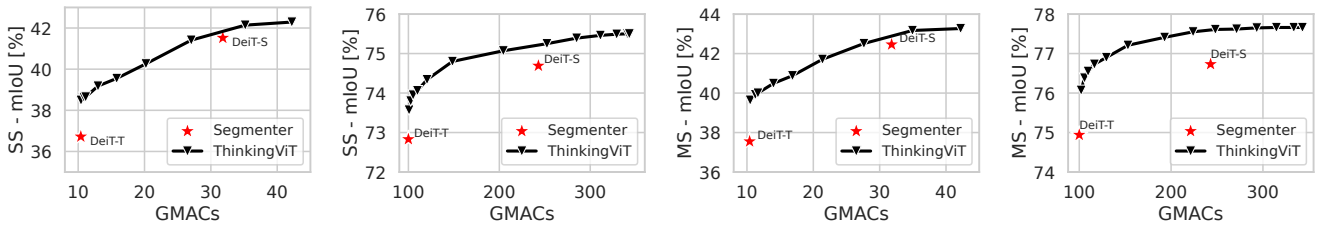
After each stage, *ThinkingViT* uses output entropy to assess certainty and decide if further computation is needed. Figure 6 shows the entropy after the first round of inference on ImageNet-A [17], ImageNet-V2 [35], and ImageNet-R [18], providing insight into how *ThinkingViT* 3H → 6H estimates uncertainty across different distribution shifts. On easier datasets like ImageNet-V2, entropy is left-skewed, reflecting confident early predictions. In contrast, harder datasets like ImageNet-A and -R produce right-skewed distributions, triggering deeper inference in *ThinkingViT*. In Figure 7, we show the load distribution on the 3H and 6H submodels under varying entropy thresholds. As entropy decreases, fewer samples proceed to the second stage, and those that exit early are rarely misclassified. This confirms that entropy effectively halts computation on easy inputs without sacrificing accuracy. Furthermore, Figure 5 shows images with different entropy levels to illustrate how *ThinkingViT* adjusts its computation. Easy images such as clear pictures of a single object have low entropy and stop after one round of thinking. More difficult images, including those with several objects, blocked views, or poor lighting, show higher entropy and trigger a second round of thinking to improve accuracy.

4.5. Semantic Segmentation

ThinkingViT builds on the vanilla ViT architecture and is therefore compatible with models designed for ViT backbones. To verify this, we replace the ViT backbone in the *Segmenter* [39] model with *ThinkingViT* and employ a linear decoder for semantic segmentation. We evaluate on the ADE20K [56] and Cityscapes [6] datasets using both single-scale and multi-scale inference. As shown in Fig. 9, *ThinkingViT* consistently outperforms DeiT-Small and DeiT-Tiny backbones, demonstrating its effectiveness as a drop-in replacement for vision transformer architectures in downstream tasks. Furthermore, Fig. 10 visualizes how *ThinkingViT* refines object boundaries and improves segmentation accuracy with additional rounds of "thinking". *ThinkingViT* also extends to other ViT based tasks such as DETR [5] style object detection, which we leave for future work.

4.6. Extension to Hierarchical Architectures

While all previous experiments were based on a vanilla ViT architecture, we also demonstrate that *ThinkingViT* can be applied to more modern variants of Vision Transformers, such as the Swin Transformer [27]. The key difference in Swin Transformer is that it consists of four stages, each with



(a) ADE20K (single-scale) (b) Cityscapes (single-scale) (c) ADE20K (multi-scale) (d) Cityscapes (multi-scale)

Figure 9. ***ThinkingViT* performance on semantic segmentation.** Comparison of mIoU versus GMACs on the ADE20K [56] and Cityscapes [6] datasets under single- and multi-scale evaluation using the Segmenter [39] pipeline. Similar to classification, *ThinkingViT* achieves a favorable trade-off between GMACs and accuracy, and outperforms the DeiT-based backbones.

an increasing number of attention heads and a decreasing number of tokens, which grow progressively larger. We integrate *ThinkingViT* on top of Swin-S, by allocating the entire first stage and half of the capacity of the last three stages for the first round, as these contain most of the layers and attention heads. The second round is then executed using the full model. As illustrated in Figure 8a, this design allows *ThinkingViT* with a Swin backbone to achieve the same accuracy as Swin-B (83.5%) while requiring only 9.5 GMACs and 50M parameters, compared to 15.4 GMACs and 88M parameters for Swin-B. Moreover, our model achieves a higher accuracy per GMAC compared to both Swin-S and Swin-T, demonstrating that our approach can improve performance even for hierarchical architectures.

4.7. Comparison with Early-Exit Models

ThinkingViT reduces computation by adapting model *width*, so its primary baselines are width-pruning approaches (Section 4.3). To additionally compare against depth-adaptive approaches, we evaluate *ThinkingViT* against several early-exit models, including SDN [24], PABEE [57], BERxiT [47], ViT-EE [1], PCEE [55], and LGViT [48]. As shown in Fig. 8b, *ThinkingViT* achieves much higher accuracy across comparable GMAC budgets. In Appendix C, we provide a detailed comparison with BranchyNet [40].

4.8. *ThinkingViT* performance on hard inputs

When the input distribution is uniformly challenging, most samples require deeper inference. To evaluate *ThinkingViT* under such conditions, we use ImageNet-R, a benchmark constructed from examples "misclassified" by ResNet-50 [16]. Since ResNet-50 has an accuracy similar to the 3H subnetwork, these samples serve as hard cases for the first stage of *ThinkingViT* and can simulate a uniformly hard input setting. Figure 6 shows that *ThinkingViT* routes most samples to the expanded 6H stage in this case. This demonstrates that *ThinkingViT*'s entropy-based routing behaves as intended. Additionally, Figure 4c shows that *ThinkingViT* still outperforms all of the baselines in these hard conditions.

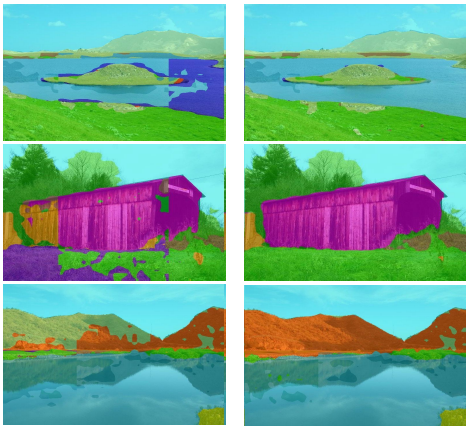


Figure 10. **ThinkingViT segmentation after one and two rounds of thinking.** Example outputs from the ADE20K [56] dataset. The second round refines object boundaries and improves segmentation quality compared to the first round.

5. Conclusion

We introduce *ThinkingViT*, a nested ViT that uses progressive thinking stages to adapt inference computation to input difficulty. The model begins with a small subset of important attention heads and halts early when confidence is high. Otherwise, within the same backbone, it activates a larger subset of attention heads and conducts a new forward pass. This iterative expansion continues until the model reaches the desired confidence level or exhausts its maximum capacity. Token Recycling further improves later stages by fusing input embeddings with features from previous stages. *ThinkingViT* outperforms both width-based and depth-based expansion strategies, and its backbone-preserving design enables seamless transfer to downstream tasks such as semantic segmentation. We also demonstrate that it extends effectively to architectures such as Swin.

Acknowledgements

This research received funding from the Federal Ministry for Digital and Transport under the CAPTN-Förde 5G project (grant no. 45FGU139H), the German Ministry of Transport and Digital Infrastructure through the CAPTN Förde Areal II project (grant no. 45DTWV08D), Federal Ministry for Economic Affairs and Energy under the CAPTN X-FERRY project (grant no. FK: 03SX612A), and the Federal Ministry for Economic Affairs and Climate Action under the Marispace-X project (grant no. 68GX21002E). It was supported in part by high-performance computing resources provided by the Kiel University Computing Centre and the Hydra computing cluster, funded by the German Research Foundation (grant no. 442268015) and the Petersen Foundation (grant no. 602157).

References

[1] Arian Bakhtiarnia, Qi Zhang, and Alexandros Iosifidis. Multi-exit vision transformer for dynamic inference. *arXiv preprint arXiv:2106.15183*, 2021. 3, 8

[2] Lucas Beyer, Olivier J. Henaff, Alexander Kolesnikov, Xiaohua Zhai, and Aaron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2002.05709*, 2020. 6

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations*, 2023. 3

[4] Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: many-in-one flexible large language model. In *Proceedings of the 41st International Conference on Machine Learning*, pages 5298–5311, 2024. 2, 3

[5] 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*, pages 213–229. Springer, 2020. 7

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 7, 8

[7] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[8] Fnu Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hanna Hajishirzi, Sham Kakade, Ali Farhadi, et al. Matformer: Nested transformer for elastic inference. *Advances in Neural Information Processing Systems*, 37:140535–140564, 2024. 1, 2, 3, 5

[9] Dujian Ding, Bicheng Xu, and Laks V. S. Lakshmanan. OC-CAM: Towards cost-efficient and accuracy-aware classification inference. In *The Thirteenth International Conference on Learning Representations*, 2025. 2, 3, 5

[10] 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. 1, 2

[11] Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, et al. Competitive programming with large reasoning models. *arXiv preprint arXiv:2502.06807*, 2025. 3

[12] Advait Gadhikar, Souptik Kumar Majumdar, Niclas Popp, Piyapat Saranrittichai, Martin Rapp, and Lukas Schott. Attention is all you need for mixture-of-depths routing. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2025. 3

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 2, 3, 4

[14] Janek Haberer, Ali Hojjat, and Olaf Landsiedel. Hydravit: Stacking heads for a scalable vit. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 1, 2, 3, 4, 5, 6

[15] Ali Hatamizadeh and Jan Kautz. Mambavision: A hybrid mamba-transformer vision backbone. *arXiv preprint arXiv:2407.08083*, 2024. 1

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 8

[17] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *arXiv preprint arXiv:1907.07174*, 2019. 6, 7

[18] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*, 2020. 6, 7

[19] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793, 2020. 1, 3, 5

[20] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017. 3

[21] Gagan Jain, Nidhi Hegde, Aditya Kusupati, Arsha Nagrani, Shyamal Buch, Prateek Jain, Anurag Arnab, and Sujoy Paul. Mixture of nested experts: Adaptive processing of visual tokens. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 3

[22] Wittawat Jitkrittum, Neha Gupta, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, and Sanjiv Kumar. When does confidence-based cascade deferral suffice? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 5

[23] Anil Kag, Igor Fedorov, Aditya Gangrade, Paul Whatmough, and Venkatesh Saligrama. Efficient edge inference by selective query. In *The Eleventh International Conference on Learning Representations*, 2023. 3, 5

[24] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR, 2019. 3, 8

[25] Abhinav Kumar, Jaechul Roh, Ali Naseh, Marzena Karpinska, Mohit Iyyer, Amir Houmansadr, and Eugene Bagdasarian. Overthink: Slowdown attacks on reasoning llms. *arXiv e-prints*, pages arXiv–2502, 2025. 3

[26] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249, 2022. 3

[27] 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. 7

[28] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019, 2022. 1

[29] 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. 3

[30] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025. 3

[31] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. 2, 3

[32] Aaron Openai: Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. 3

[33] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021. 3

[34] David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024. 3

[35] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400, 2019. 6, 7

[36] Florence Regol, Joud Chataoui, and Mark Coates. Jointly-learned exit and inference for a dynamic neural network: Jei-dnn. *arXiv preprint arXiv:2310.09163*, 2023. 3

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. 5

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 2, 3, 4

[39] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021. 7, 8

[40] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016. 3, 8, 1

[41] 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. 1, 5

[42] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *European conference on computer vision*, pages 516–533. Springer, 2022. 1

[43] Mojtaba Valipour, Mehdi Rezagholizadeh, Hossein Rajabzadeh, Marzieh Tahaei, Boxing Chen, and Ali Ghodsi. Sortednet, a place for every network and every network in its place: Towards a generalized solution for training many-in-one neural networks. *arXiv preprint arXiv:2309.00255*, 2023. 1, 3, 5

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3

[45] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019. 6

[46] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 5

[47] Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. Berxit: Early exiting for bert with better fine-tuning and extension to

regression. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: Main Volume*, pages 91–104, 2021. 3, 8

[48] Guanyu Xu, Jiawei Hao, Li Shen, Han Hu, Yong Luo, Hui Lin, and Jialie Shen. Lgvit: Dynamic early exiting for accelerating vision transformer. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 9103–9114, 2023. 3, 8

[49] Guanyu Xu, Zhiwei Hao, Yong Luo, Han Hu, Jianping An, and Shiwen Mao. Vision transformers on the edge: A comprehensive survey of model compression and acceleration strategies. *arXiv preprint arXiv:2503.02891*, 2025. 2

[50] 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. 3

[51] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1803–1811, 2019. 3, 4

[52] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018. 3

[53] Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? *arXiv preprint arXiv:2502.12215*, 2025. 3

[54] Yitian Zhang, Huseyin Coskun, Xu Ma, Huan Wang, Ke Ma, Stephen Xi Chen, Derek Hao Hu, and Yun Fu. Slicing vision transformer for flexibile inference. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2, 3

[55] Zhen Zhang, Wei Zhu, Jinfan Zhang, Peng Wang, Rize Jin, and Tae-Sun Chung. Pcee-bert: Accelerating bert inference via patient and confident early exiting. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 327–338, 2022. 3, 8

[56] 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. 7, 8

[57] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341, 2020. 3, 8

[58] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022. 2, 3

ThinkingViT: Matryoshka Thinking Vision Transformer for Elastic Inference

Supplementary Material

A. Adaptive inference performance of *ThinkingViT* under varying entropy thresholds

Table 3 presents detailed results of *ThinkingViT*'s adaptive inference behavior under different entropy thresholds. We report accuracy, number of parameters, throughput, total compute (in GMACs), and the ratio of inputs that proceed to the second stage of inference of *ThinkingViT* 3H $\rightarrow$ 6H.

B. Results on ImageNet variants

Figure 12 shows the performance of *ThinkingViT* on ImageNet variants. Note that the GMACs for ImageNet-R and ImageNet-V2 are reported in Figure 4c and Figure 4b, respectively.

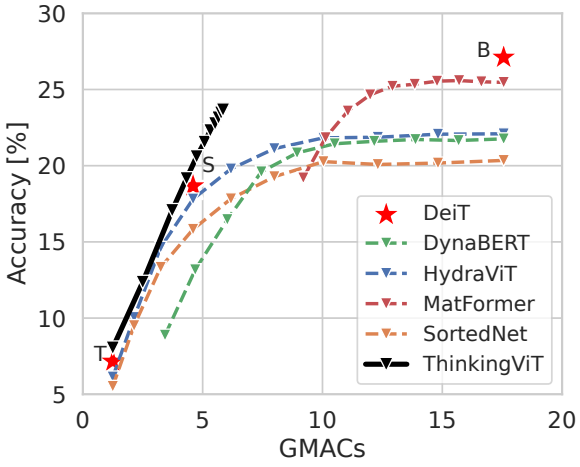


Figure 11. Full results of *ThinkingViT* and baselines in terms of GMACs on ImageNet-A.

C. Comparison with BranchyNet

Table 2 compares *ThinkingViT* with BranchyNet, a standard early exit model[40], applied to DeiT [42] that matches its GMACs and throughput. The baseline consists of 24 layers: the first 12 use 3 attention heads (3H) and the remaining 12 use 6 heads (6H). Similar to *ThinkingViT*, the exits are jointly trained together. At the 3H point, *ThinkingViT* records slightly lower accuracy because the first three heads must generate representations that remain compatible with the later expansion to six heads. Additionally, their weights must be trained in a way that allows the weights of the subsequent three heads to build upon them.

After expanding to 6 attention heads, *ThinkingViT* reaches 81.44% top-1 accuracy, improving by 3.37 p.p. upon the early exit baseline, which attains 78.08%. This improvement likely stems from *ThinkingViT* increasing the number of active attention heads within a fixed 12-layer backbone, allowing gradients to pass through fewer layers (12 layers compared to 24 layers) and reducing some of the optimization challenges associated with deeper networks. These findings indicate that allocating compute along the width dimension by gradually increasing attention heads can be more effective than depth-based early exit strategies used in prior work. Further, since it adjusts model width rather than depth, *ThinkingViT* is complementary to early-exit strategies and can be combined with them to further expand deployment flexibility.

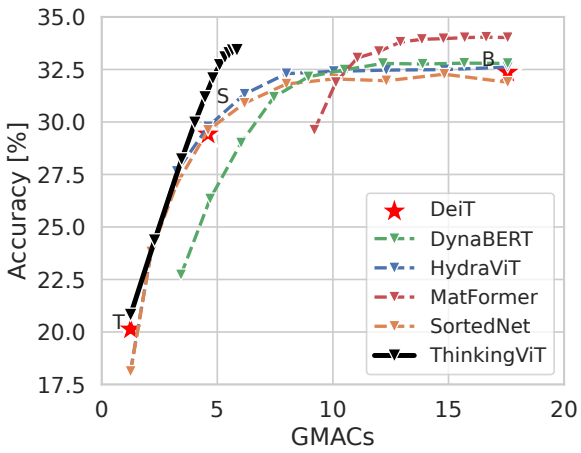


Figure 12. Full results of *ThinkingViT* and baselines in terms of GMACs on ImageNet-Sketch.

D. Ablation of different token recycling strategies

In Table 4, we compare several design variants for recycling information from the first round of inference to get better performance in the second round. We conduct these experiments on *ThinkingViT* 3H $\rightarrow$ 6H. In the *Layerwise Activation Snapshots* variant, we cache the hidden states at each of the 12 layers from the first round and feed them into the corresponding layers in the second round. However, this approach performs suboptimally, likely because it forces all intermediate representations from the first round to be reused in the second round, which reduces the performance. In the *Memory Tokens* design, we introduce a set of learnable memory tokens [7] that, in the first round, store helpful information for the second round. We also

Table 2. Comparison of *ThinkingViT* and BranchyNet on DeiT.

Model	GMACs	Throughput [#]	Params [M]	Accuracy [%]
DeiT-Tiny	1.25	10047.6	5.7	72.2
DeiT-Small	4.6	4603.6	22.1	79.9
BranchyNet 3H	1.25	10047.6	27.8	74.51
BranchyNet 3H → 6H	5.85	3157.1	27.8	78.08
***ThinkingViT* 3H**	1.25	10047.6	22.1	73.58
***ThinkingViT* 3H → 6H**	5.85	3157.1	22.1	**81.44 (+3.36)**

Table 3. Performance metrics of *ThinkingViT* 3H → 6H across different entropy thresholds

Entropy Threshold	Accuracy	Throughput [#/s]	Params [M]	GMACs	Second Round Call Ratio [%]
0	81.444	3157.09	22.01	5.85	100.0
0.1	81.440	3347.69	22.01	5.47	71.7
0.3	81.438	3955.05	22.01	4.50	70.58
0.5	81.386	4380.71	22.01	3.98	59.29
0.7	81.230	4807.04	22.01	3.55	49.95
0.9	80.714	5342.47	22.01	3.11	40.36
1.1	79.990	5918.90	22.01	2.72	31.97
1.3	79.114	6535.13	22.01	2.38	24.63
1.5	77.936	7201.46	22.01	2.08	18.11
1.7	76.766	7944.38	22.01	1.81	12.13
2	74.736	9203.90	22.01	1.44	4.20
2.5	73.580	10047.60	22.01	1.25	0.0

experiment with hybrid strategies that combine activation snapshots with Memory Tokens or introduce a fresh `[CLS]` token in the second round. We additionally evaluated the use of KV-cache reuse [44], originally introduced for generative models, and include the results in Table 4. Ultimately, we find that the simplest strategy, *Final-Layer Token Recycling*, which reuses the features from the last layer of the first round, achieves the best performance. This indicates that the output of the first round already captures sufficient high-level information to guide the second round effectively, while being comparatively simple to implement. It is important to note that except the KV-cache experiment, all results were conducted during the early experimental phase of *ThinkingViT*, and due to training resource constraints, they were not trained with the full joint training strategy described in Section 3.3, but rather with the stochastic training method introduced in [14]. As a result, their accuracies are slightly lower than those of the final model reported in Section 4.

E. Ablation of different fusing strategies

Table 5 compares six fusion strategies that recycle the tokens produced in the first round, denoted as z_1, by incorporating them into the initial patch embeddings of the second round, $\mathcal{E}_2$, on *ThinkingViT* with a configuration of $3H \rightarrow 6H$. Each variant forms the second-round input as a linear blend, where α is a learnable scalar. The strategies differ in how the two embeddings are projected to the same dimensional space. We consider two projection strategies: (I) a parameter-free approach that either repeats the embedding or pads the lower-dimensional embeddings with zeros, and (II) a learnable linear projection. Empirically, the learnable projection achieves the best overall accuracy by having the highest accuracy in the second round while maintaining performance comparable to the strongest model configuration in the first round. In addition, we evaluate different initializations of α and observe that initializing with $\alpha = 0$ consistently leads to the best performance. This initialization allows the model to begin training without relying on information from the first round, thereby enabling it to learn the optimal degree of reuse. For example, in *ThinkingViT* configured with 3H → 6H heads, the learned value of α converges to -0.19.

F. Prediction dynamics across two inference rounds on ImageNet-1K

Figure 13 presents prediction dynamics across two inference rounds of *ThinkingViT* 3H → 6H on the ImageNet-1K validation set. A small portion of samples, approximately 2%, were correctly classified in the first round but misclassified in the second. This phenomenon, often attributed to *over-*

Table 4. Comparison of design variants for conditioning the second round of inference on the first.

Design Variant	First Thought Acc. [%]	Second Thought Acc. [%]
DeiT Baseline (Tiny → Small)	72.2	79.9
Layerwise Activation Snapshots	73.794	78.566
Memory Tokens	73.96	78.9
Layerwise Activation Snapshots + new `[CLS]` in second round	73.58	77.94
Layerwise Activation Snapshots + Memory Tokens	73.71	79.04
KV-Caching [44]	73.872	75.674
Final-Layer Token Recycling (*ThinkingViT*)	73.13	**80.05**

Table 5. Impact of different fusion strategies for integrating first-round tokens (z_1) into second-round embeddings ($\mathcal{E}_2$) during progressive inference on *ThinkingViT* with 3H → 6H.

Fusion Method	Dim Alignment	Note	Acc. [%] 1st Round	Acc. [%] 2nd Round
$\alpha \cdot z_1 + \mathcal{E}_2$	Pad z_1 with zeros	Pad the first half, $init(\alpha) = 0$	73.32%	81.37%
$\alpha \cdot z_1 + \mathcal{E}_2$	Pad z_1 with zeros	Pad the second half, $init(\alpha) = 0$	**74.12%**	80.32%
$\alpha \cdot z_1 + \mathcal{E}_2$	Repeat z_1	$init(\alpha) = 1$	72.99%	80.87%
$\alpha \cdot z_1 + \mathcal{E}_2$	Repeat z_1	$init(\alpha) = 0$	73.14%	81.41%
$z_1 + \alpha \cdot \mathcal{E}_2$	Repeat z_1	$init(\alpha) = 0$	72.89%	80.51%
$\alpha \cdot z_1 + \mathcal{E}_2$	Linear (*ThinkingViT*)	$init(\alpha) = 0$	73.58%	**81.44%**

thinking [25], is also observed in other architectures such as ResNet and Swin Transformers [9], where smaller models sometimes outperform larger ones on few samples by avoiding unnecessary complexity. The majority of samples, around 70%, were correctly classified in both rounds. These generally correspond to visually simple or unambiguous cases where one round of inference is sufficient. Roughly 10% of samples were initially misclassified but corrected in the second round, demonstrating the benefit of additional reasoning for harder examples. Finally, around 16% remained misclassified across both rounds, indicating that these inputs are intrinsically ambiguous or fall outside the model's capacity, even with increased computation.

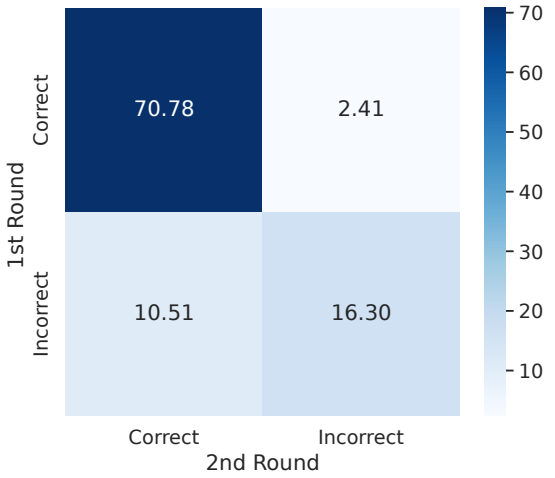


Figure 13. Prediction dynamics of *ThinkingViT* 3H → 6H across two inference rounds on ImageNet-1K.

G. Comparing and combining *ThinkingViT* with other adaptive baselines

ThinkingViT builds upon a nested backbone to enable input adaptivity, so our evaluation in Section 4 focuses on nested Transformer baselines. For completeness, we also evaluate against several representative token-level dynamic models, including MoD [34], AMoD [12], ToMe [3], A-ViT [50], AdaViT [29], and DynamicViT [33]. As shown in Figure 14, *ThinkingViT* achieves greater scalability and consistently better GMACs-Accuracy tradeoffs.

We note that *ThinkingViT* performs routing at the *image level*, which is orthogonal to token-level pruning, and thus can be combined with such approaches to provide additional flexibility. To demonstrate this, we incorporate the token pruning mechanism of DynamicViT [33] into the second round of *ThinkingViT* 3H → 6H, applying a pruning ratio of 0.8. As shown in Fig. 14, integrating DynamicViT further improves both efficiency and accuracy, indicating that *ThinkingViT* 3H → 6H is compatible with and complementary to existing token pruning approaches.

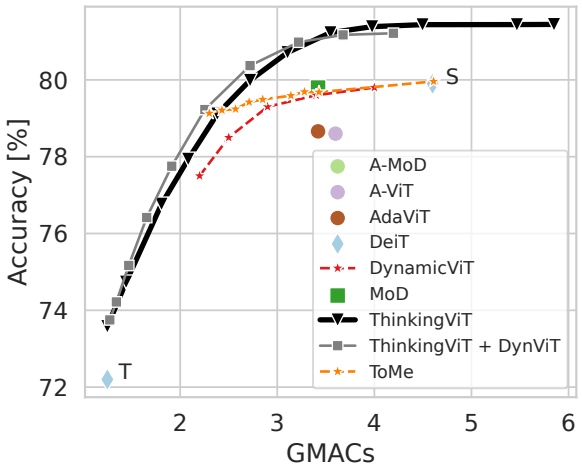


Figure 14. Comparing and combining *ThinkingViT* with token pruning baselines.

H. Comparison of accuracy vs. GMACs for baselines based on DeiT-Small

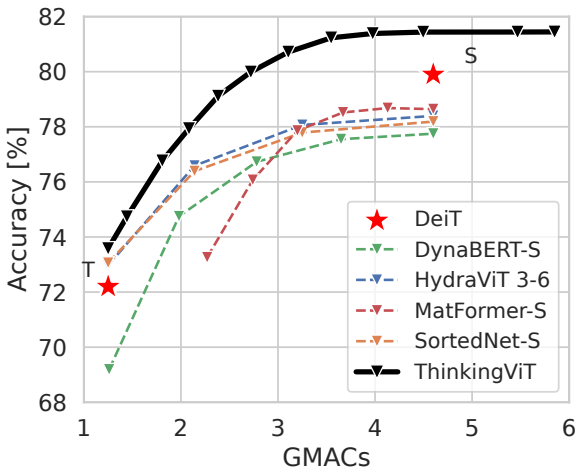


Figure 15. Accuracy versus GMACs on the ImageNet-1K validation set. All baseline models are based on DeiT-Small. *ThinkingViT* consistently outperforms these baselines by achieving higher accuracy at comparable or lower computational cost.

In Figure 15, we compare accuracy versus GMACs on ImageNet-1K using DeiT-Small as the common backbone for all baselines. *ThinkingViT* achieves higher accuracy at similar or lower compute budgets, showing a consistently more favorable tradeoff. This demonstrates that the benefits of *ThinkingViT* hold across backbone scales.

I. Limitations

Training Overhead: Compared to training multiple standalone models, nested models like *ThinkingViT* incur higher training costs to reach similar performance levels across all stages. This is a known limitation of nested architectures, which share parameters and require joint optimization to maintain accuracy at varying compute levels.

Jumping Too Far Reduces Effectiveness: When the model transitions from a very small subset of attention heads (e.g., 3H) directly to a full configuration (e.g., 12H) in the second stage, performance suffers compared to using an intermediate stage (e.g., 9H) instead. This suggests that overly aggressive compute expansion can disrupt representational continuity, emphasizing the importance of smooth progression in staged inference.

J. Batch inference

After each "thinking" round, we filter out the samples whose entropy falls below the stopping threshold (i.e., confident predictions). The remaining uncertain samples are then rebatched and forwarded to the next round. Since all rounds share the same Transformer backbone and Token Recycling only adjusts the input embeddings, no additional batch scheduling is needed. As a result, *ThinkingViT* integrates smoothly into standard batched inference engines, where the model proceeds in synchronized rounds and the batch size gradually decreases as samples exit early; see Algorithm 1.

Algorithm 1 Batched Inference with *ThinkingViT*

$\mathcal{B} \leftarrow$ input batch
for $r = 1$ to R **do** ▷ Progressive "thinking" rounds
  $\hat{y} \leftarrow \text{Forward}(\mathcal{B}, r)$
  $\mathcal{C} \leftarrow \{i \in \mathcal{B} \mid \text{Entropy}(\hat{y}_i) \leq \tau\}$ ▷ Confident samples
  $\mathcal{U} \leftarrow \mathcal{B} \setminus \mathcal{C}$ ▷ Uncertain samples
  output predictions for samples in $\mathcal{C}$
  if $\mathcal{U}$ is empty **then break**
  end if
  $\mathcal{B} \leftarrow \text{Rebatch}(\mathcal{U})$ ▷ Shrink batch
end for

K. Effect of attention-head expansion and loss weighting on multi-round thinking

Table 6 analyzes the effect of progressively increasing the attention-head capacity (e.g., 3H→6H, 3H→9H, 3H→12H) and varying loss-weighting schemes between the first and second rounds of thinking. Starting with 3 heads and expanding to 6 (3H → 6H) yields the best first-round accuracy, while 3H → 9H achieves the highest second-round accuracy on ImageNet-1K. Notably, by using loss weighting we can

Table 6. Impact of attention expansion and loss weighting on *ThinkingViT* with two rounds of progressive thinking on ImageNet-1K.

Model Variant	**Loss Weight**	**Acc. [%]**	
		1st Round	**2nd Round**
ThinkingViT 2H → 3H	[0.5, 0.5]	65.35	74.13
ThinkingViT 3H → 6H	[0.5, 0.5]	73.58	81.44
ThinkingViT 3H → 6H	[0.4, 0.6]	73.22	81.43
ThinkingViT 3H → 6H	[0.6, 0.4]	73.93	81.28
ThinkingViT 3H → 9H	[0.5, 0.5]	72.51	82.02
ThinkingViT 3H → 9H	[0.4, 0.6]	71.71	82.15
ThinkingViT 3H → 9H	[0.6, 0.4]	73.02	81.92
ThinkingViT 3H → 12H	[0.5, 0.5]	72.03	81.70
ThinkingViT 3H → 12H	[0.4, 0.6]	70.78	81.80
ThinkingViT 3H → 12H	[0.6, 0.4]	72.23	81.51

tune the model to put more focus on the first round or second round based on the desired trade-off between computation and accuracy. We also explore 3-stage variants in Table 7, demonstrating that *ThinkingViT* naturally scales to deeper thinking hierarchies and yields higher final accuracy. See Figure 16 for a high-resolution plot of *ThinkingViT* variants.

Table 7. *ThinkingViT* performance with three rounds of thinking on ImageNet-1K, demonstrating that *ThinkingViT* naturally scales to deeper thinking hierarchies and yields higher final accuracy.

Model	Acc. [%] 1st Round	2nd Round	3rd Round
ThinkingViT 2H (33%) → 3H (50%) → 6H (100%)	64.62	73.56	81.43
ThinkingViT 3H (25%) → 6H (50%) → 12H (100%)	70.77	80.00	82.35

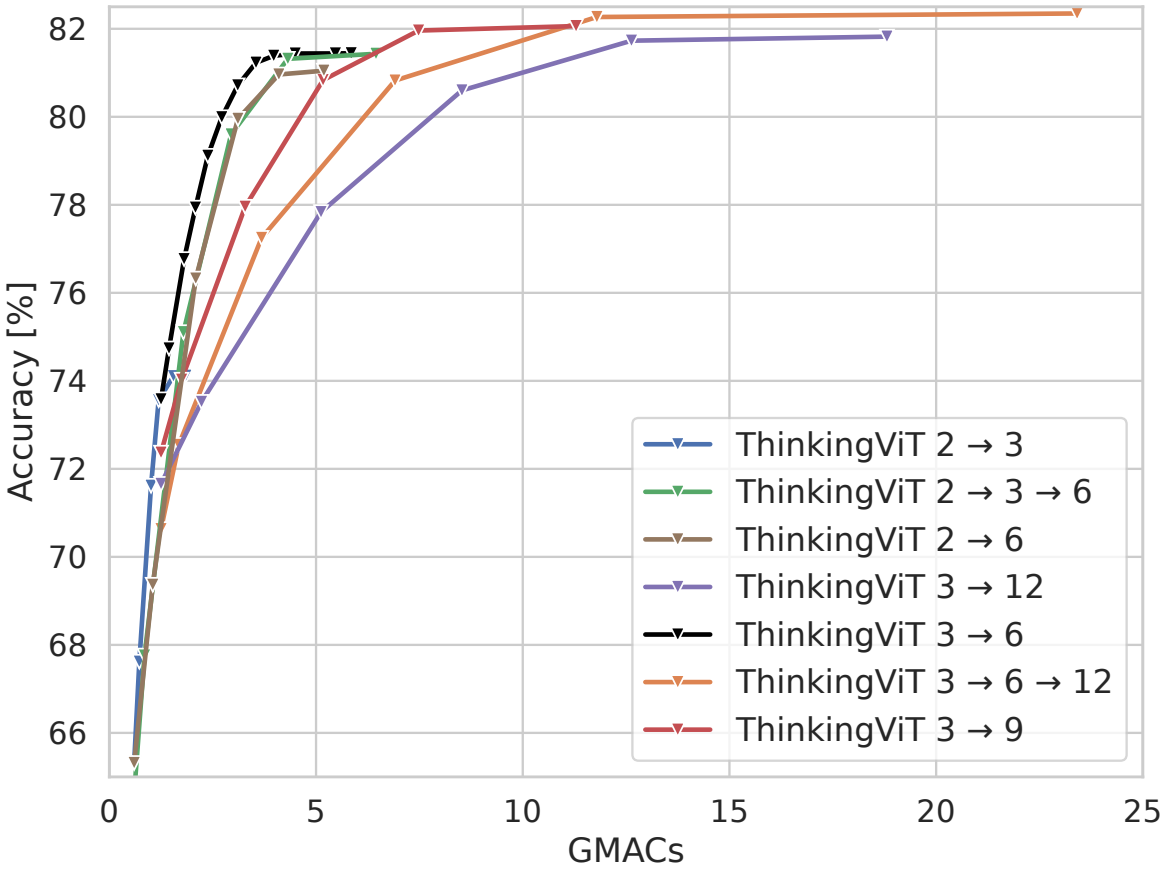


Figure 16. A higher-resolution version of Figure 3.