# IS DEPTH ALL YOU NEED? RETHINKING VIT CAPACITY VIA STRUCTURAL RE-PARAMETERIZATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Vision Transformers (ViTs) are computationally expensive largely due to their deeply stacked layer designs, yet prevailing efficiency research has concentrated on algorithmic-level strategies such as token pruning and attention mechanism acceleration. This raises a fundamental but underexplored architectural question: is it possible to decrease ViT depth without sacrificing model capacity? We address this question by introducing a branch-based structural re-parameterization framework that augments transformer blocks with parallel branches during training, thereby enhancing optimization flexibility and model capacity. After training, these branches are systematically consolidated into a single linear pathway through exact re-parameterization applied at the inputs of nonlinear activations, ensuring strict functional equivalence between the expanded and collapsed models. Experiments across multiple ViT backbones demonstrate that this approach achieves a 2–4 $\times$ reduction in network depth while maintaining competitive classification performance on ImageNet-1K. Beyond depth reduction, we further demonstrate that re-parameterization functions as a general capacity enhancer, boosting the accuracy of iso-depth ViTs and improving vision encoders within vision-language models (VLMs). These findings indicate that heavily stacked transformer architectures may be redundant for error-tolerant and resource-constrained deployment scenarios, opening promising avenues for the design of efficient vision transformers.

## 1 INTRODUCTION

Vision Transformers (ViTs) have achieved state-of-the-art performance across many vision tasks by modeling long-range dependencies (Dosovitskiy et al., 2020), yet their quadratic attention and deep sequential architectures impose significant deployment challenges on resource-constrained platforms. This limitation is particularly acute for edge and mobile devices, where strict latency, memory, and power budgets favor shallow and highly parallel models. Even compact ViTs remain difficult to deploy efficiently on general-purpose ARM processors and mobile CPUs. For instance, TinyViT (Wu et al., 2022), among the smallest ImageNet-1K ViTs, still employs 12 transformer layers and 5.7M parameters, leading to substantial sequential latency despite its modest size. While pruning and quantization can reduce storage and computational overhead, transformer depth continues to bottleneck real-time execution on embedded platforms (Lee et al., 2019; Li et al., 2022b; Wu et al., 2022). These observations motivate a shift toward ultra-shallow architectures that minimize sequential computation while preserving representational power.

Beyond mobile deployment, shallow architectures are increasingly critical for emerging platforms such as analog in-memory and photonic computing (Negi et al., 2025; Khilo et al., 2012). In these systems, data conversion and noise accumulation scale with network depth, with analog-to-digital converter (ADCs) alone accounting for up to 60% of energy and 80% of area in compute-in-memory accelerators (Saxena et al., 2022), while deeper analog networks suffer progressive accuracy degradation (Wan et al., 2022). Photonic processors similarly incur costly optical–electrical conversions at each layer (Shen et al., 2017). These constraints favor ultra-shallow models (4–6 layers) that balance efficiency and expressivity. Yet most ViT efficiency efforts focus on attention optimization or token reduction (Wang et al., 2021; Rao et al., 2021), implicitly assuming deep architectures, rather than fundamentally rethinking depth itself. Recent work shows that parallel architectures can match deep networks while reducing sequential computation (Goyal et al., 2022), and structural re-parameterization enables overparameterized training models to collapse into efficient inference architectures (Ding et al., 2021b). While these ideas have been explored primarily in



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

convolutional neural networks (CNNs), they motivate a new direction for ViTs: combining parallel training with deployment-oriented re-parameterization to drastically reduce depth while preserving representational capacity under fixed inference budgets. *To the best of our knowledge, this work represents the first attempt to demonstrate competitive ViT performance with $\leq$6 layers, while showing that re-parameterization can act as an orthogonal capacity booster at iso-depth.*

Extending existing efficiency approaches to such extreme depth reduction is non-trivial and often counterproductive. Knowledge distillation methods, while effective for moderate compression (Touvron et al., 2021a; Hao et al., 2022), face severe limitations when applied to ultra-shallow students. Recent distillation work shows that DeiT-Tiny students with 12 layers can achieve at most 76.5% ImageNet accuracy even with sophisticated manifold distillation (Hao et al., 2022), and performance degrades rapidly as student depth decreases below 8-10 layers (Chen et al., 2022). Traditional pruning approaches (Kwon et al., 2022; Liu et al., 2024) are designed for moderate depth reduction (removing 20-30% of layers) and cannot handle the extreme compression ratios required for 6-layer models without catastrophic accuracy loss. Other depth compression approaches, such as network overparameterization (Yu et al., 2025) and non-linearity manipulation (Fu et al., 2022; Bhardwaj et al., 2022), have not yet demonstrated scalability to ViTs. Structural modifications such as depth-wise convolutions (Zhang et al., 2025) and hybrid CNN–ViT designs require fundamental architectural changes, limiting compatibility with standard ViT deployments and pretrained vision backbones widely used in vision systems.

**Our Contributions.** We propose a *progressive structural re-parameterization framework for efficient ViTs*. Our method trains parallel-branch transformer blocks that are gradually merged during training and ultimately collapse into standard single-path architectures for lightweight inference. By joining branches at the inputs of key nonlinear activations, including softmax and GeLU, we enable exact algebraic re-parameterization without approximation loss. This allows branches to diversify early in training while converging to a compact representation later. Using this approach, we demonstrate that non-deep ViTs with only 6 layers achieve competitive ImageNet-1K accuracy. We further show that the same re-parameterization mechanism acts as a *training-time capacity booster*, improving the accuracy of iso-depth ViTs and pretrained vision encoders without increasing inference complexity. Our results challenge the assumption that sequential depth is essential for ViT performance and open new opportunities for deployment on mobile, edge, and analog hardware platforms.

## 2 Related Work

**ViT Efficiency:** Research on efficient ViTs has explored multiple levels of optimization. *Token-level methods*, such as DynamicViT (Rao et al., 2021) and TokenLearner (Ryoo et al., 2021), reduce the number of tokens processed through dynamic selection or learned aggregation. *Attention-level approaches* address the quadratic cost of self-attention using linear variants (Katharopoulos et al., 2020), sparse patterns (Child et al., 2019), or multi-scale designs (Wang et al., 2021). Recent architectures challenge core transformer assumptions: EfficientFormer (Li et al., 2022b) achieves mobile-speed inference with dimension-consistent design, SwiftFormer (Shaker et al., 2023) replaces key-value matrix multiplication with element-wise operations for linear complexity, SHViT (Yun & Ro, 2024) uses single-head attention for speed, and MoR-ViT (Li, 2025) applies mixture-of-recursions for dynamic depth allocation. Hybrid CNN-transformer models combine local convolutional inductive biases with global modeling. Examples include CvT (Wu et al., 2021), CMT (Guo et al., 2022), MobileViT (Mehta & Rastegari, 2022), and NextViT (Li et al., 2022a), which strategically merge CNN and transformer blocks for better accuracy-efficiency trade-offs. Another major direction for efficiency is pruning and slimming of ViTs, which focuses on identifying redundant components to reduce FLOPs and parameters. Self-Slimmed ViT (Zong et al., 2022) prunes channels and attention heads using sparsity-inducing gates, while Unified Visual Transformer Compression (Yu et al., 2022) jointly optimizes pruning, quantization, and knowledge distillation in a single framework. X-Pruner (Yu & Xiang, 2023) introduces a differentiable search process to prune layers, heads, and tokens simultaneously, and LPViT (Xu et al., 2024) leverages frequency-domain low-pass filtering to reduce computational cost. While these methods reduce *within-layer* computation or partially remove layers, they rely on approximations or post-hoc distillation and remain tied to deep sequential architectures.

**Structural Re-parameterization:** Structural re-parameterization decouples training-time architectural complexity from inference-time efficiency. RepVGG (Ding et al., 2021b) pioneered this idea by showing that multi-branch training structures can be algebraically collapsed into single-branch inference networks without performance loss. Diverse Branch Block (DBB) (Ding et al., 2021a)

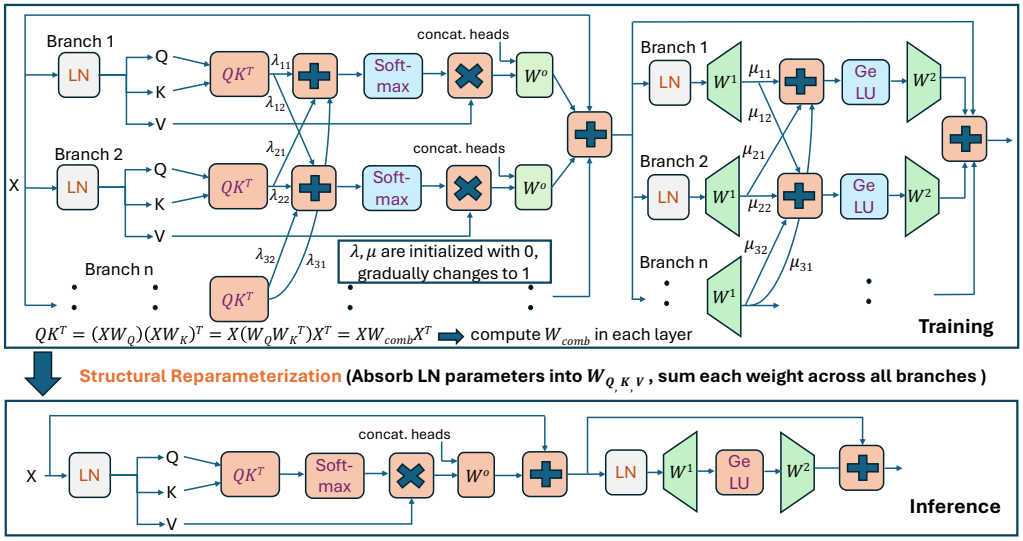



Figure 1: Overview of training-time multi-branch ViTs and inference-time collapsed architecture.

extended this with multi-scale transformations during training, while OREPA (Hu et al., 2022) introduced online re-parameterization to reduce training-time memory costs. UniRepLKNet (Ding et al., 2024) showed that extremely large kernels can be efficiently compressed for deployment, and RepViT (Wang et al., 2024) adapted these ideas to mobile backbones such as MobileNetV3. Extensions to transformers have been limited: FastViT (Vasu et al., 2023) introduced RepMixer for token mixing, while RePaViT (Xu et al., 2025) applied re-parameterization only to FFN layers. SLAB (Guo et al., 2024) introduced simplified linear attention with re-parameterized BatchNorm for efficient transformers, but focused on attention and normalization redesign rather than depth re-parameterization or collapsing ViTs into shallow models.

# 3 Proposed Method

## 3.1 Overview and Challenges

ViTs achieve strong performance through deep stacks of transformer blocks, each containing multi-head self-attention (MHSA) and feed-forward network (FFN) modules. However, this depth creates severe computational bottlenecks for deployment. Our approach mitigates this by training parallel transformer branches that are later collapsed into a single branch at inference, drastically reducing depth while preserving representational capacity. The key idea is that, with proper constraints, parallel branches can be *algebraically re-parameterized* into one equivalent branch without approximation error. Consider two parallel transformer blocks processing the same input $X$. Each branch has independent parameters and computes its own transformations, but we seek to merge them into a single set of operations. If the blocks contained only linear operations, this would be trivial, and we could simply sum the weight matrices. In practice, three challenges prevent naive re-parameterization: *First, non-linearities*, softmax in MHSA and GELU in FFN, break the linearity needed for direct combination. For example, $\texttt{softmax}(QK_A^T) + \texttt{softmax}(QK_B^T) \neq \texttt{softmax}(QK_A^T + QK_B^T)$. *Second, Layer Normalization (LN)* within each block introduces mismatched normalization statistics across branches, disrupting fusion. *Third, multi-head attention concatenates head outputs*, making it impossible to directly align and merge branches since $[H_{A,1}; \ldots; H_{A,h}]$ from Branch A cannot be cleanly combined with Branch B.

We address these challenges through a unified framework that combines architectural modifications with progressive training. LN operations are extracted from parallel branches, and MHSA is re-formulated using blockwise operations to replace concatenation with summation. During training, we progressively join branch outputs at the inputs to non-linear functions, so that by the end of training, all branches receive identical inputs. This guarantees exact algebraic re-parameterization while leveraging the diversity and capacity benefits of parallel training.

## 3.2 Computational Reformulation for Re-parameterization

**Layer Normalization Factorization**: To enable structural re-parameterization, we must first address the fundamental incompatibility created by LN operations within parallel branches. The standard

ViT architecture applies LN before each sub-module: one before MHSA and one before FFN. In a parallel-branch setting, if these normalizations occur within branches, each branch computes different normalization statistics, preventing algebraic combination. Our solution restructures the architecture by extracting LN operations from within the parallel branches and positioning them before the branching points. This ensures both branches receive identically normalized inputs. To maintain mathematical equivalence with this restructuring, we absorb the LN parameters into adjacent linear transformations.

*For FFN modules*, we absorb the LN parameters forward into the FFN's first linear transformation. This is possible because the FFN begins with a linear layer that can directly incorporate the normalization's scale and bias parameters. The original computation $W_1 \cdot \text{LN}(x)$ becomes: $W_1^{\text{new}} = W_1 \cdot \text{diag}(\gamma), \quad b_1^{\text{new}} = W_1\beta + b_1$, where $\text{diag}(\gamma)$ creates a diagonal matrix from the scale parameters. This absorption maintains the mathematical equivalence while eliminating LN from within the parallel branches. *For MHSA modules*, forward absorption is not feasible because it disrupts the algebraic structure required for attention re-parameterization. Specifically, attention relies on the $XWX^T$ formulation (detailed in Section 3.3), and inserting LN parameters would break this property. Instead, we *factorize* this LN operator across branches by applying a single shared normalization before the branch split, so that all branches receive identical normalized inputs. This is mathematically equivalent to duplicating the same LN inside each branch, but avoids invalid absorption across residual connections while preserving exact equivalence to the standard transformer block. For the first MHSA layer, this shared LN is applied directly after the patch embedding.

**Blockwise Multi-Head Attention Reformulation**: The standard multi-head attention implementation creates a structural barrier to re-parameterization through its use of concatenation operations. After computing attention for each head independently, the outputs are concatenated before the final projection, which prevents clean algebraic combination of parallel branches. Our solution reformulates multi-head attention using blockwise operations that eliminate concatenation entirely. Instead of concatenating head outputs before applying a monolithic output projection matrix $W^O \in \mathbb{R}^{d\times d}$, we partition this matrix into head-specific blocks: $W^O = [W_1^O, W_2^O, \ldots, W_h^O], \quad$ where $W_i^O \in \mathbb{R}^{d_h\times d}$. Each head's output is now independently projected and the final output is computed through summation rather than concatenation followed by projection: $\texttt{Out} = \sum_{i=1}^{h} H_i W_i^O = \sum_{i=1}^{h} \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i W_i^O$. This reformulation is mathematically equivalent to standard multi-head attention but replaces the non-linear concatenation with linear summation, enabling clean re-parameterization after training. This blockwise computation introduces no additional computational overhead, as modern GPUs and edge accelerators can efficiently execute these head-wise operations in parallel through native block matrix multiplication primitives, eliminating any synchronization barriers that might arise from the summation operation.

### 3.3 Progressive Joining Framework

With the architectural modifications in place, we now present our progressive joining mechanism that enables structural re-parameterization while maintaining diverse representations during training. The key innovation lies in how we handle the computation before and after non-linear functions differently to enable algebraic combination.

**Re-parameterization Before Softmax**: For attention, the key idea is to reorder the score computation to enable pre-softmax fusion across branches. In standard attention, each branch computes $QK^T = (XW^Q)(XW^K)^T$. We rewrite this as $QK^T = XW^Q{W^K}^T X^T = XWX^T$, where $W = W^Q{W^K}^T \in \mathbb{R}^{d\times d}$ is a combined weight matrix (with $W^Q, W^K \in \mathbb{R}^{d\times d_h}$). This reformulation is crucial because it expresses attention scores in the unified form $XWX^T$, allowing algebraic combination of weights across branches prior to softmax. Computing $XWX^T$ operates in the full embedding dimension $d$, rather than the per-head dimension $d_h$, and is therefore more expensive than conventional $QK^T$. For DeiT-Tiny ($d$=192, $n$=197 tokens), this requires 58.8M operations versus 43.9M for standard attention (a 33% increase). However, the combined matrix $W$ can be precomputed offline, reducing runtime dataflow from three matrix multiplications ($X\to Q$, $X\to K$, $QK^T$) to two ($X\to XW$, $XWX^T$), thereby improving efficiency. In practice, this overhead is minor, as FFNs dominate computation (116.1M FFN vs. 43.9M MHSA operations in DeiT-Tiny), resulting in only $\sim$7% total compute overhead. A similar $\sim$7% increase is observed in the number of trainable parameters, and across diverse ViT backbones we observe no more than 8% overhead.

Consider two branches, $A$ and $B$. During training with the progressive joining parameter $\lambda(t)$,

$$\text{A \& B attention scores: } XW_AX^T, XW_BX^T$$
$$\text{A \& B softmax inputs: } X(W_A{+}\lambda(t)W_B)X^T, X(W_B{+}\lambda(t)W_A)X^T$$

One challenge of joining is scale amplification: merging multiple score distributions increases variance and can destabilize training. Assuming $W_A$ and $W_B$ are independently and identically distributed, the effective variance of the combined attention logits grows with $(1{+}\lambda^2(t))$. Thus, the rectified pre-softmax scale becomes $(1{+}\lambda^2(t))\sqrt{d_k}$. $\lambda(t)$ follows a linear schedule: $\lambda(t){=}t$, where $t$ is uniformly varied from 0 to 1 during training. This schedule ensures smooth transition from independent branch processing ($\lambda{=}0$) to fully joined computation ($\lambda{=}1$), allowing branches to develop complementary features while gradually aligning their computations for perfect re-parameterization. At convergence when $\lambda{=}1$, both branches receive $X(W_A{+}W_B)X^T$, enabling perfect re-parameterization with $W_{\text{combined}}{=}W_A{+}W_B{=}W_A^QW_A^{K^T}{+}W_B^QW_B^{K^T}$.

**Re-parameterization After Softmax**: After the softmax operation, both branches produce identical MHSA weights when $\lambda{=}1$. For each head $i$, the MHSA computation is

$$H_i{=}\texttt{softmax}\left(\frac{X(W_{A,i}^QW_{A,i}^{K^T}+W_{B,i}^QW_{B,i}^{K^T})X^T}{2\sqrt{d_k}}\right)\cdot X(W_{A,i}^V+W_{B,i}^V)$$

where $H_i$ is the combined attention output for head $i$. Since both branches now compute identical softmax outputs for each head, the value projections can be combined as $W_{i,\text{combined}}^V{=}W_{A,i}^V{+}W_{B,i}^V$. The final output across all heads, using our blockwise reformulation is

$$\sum_{i=1}^{h} H_i \cdot W_{i,\text{combined}}^O = \sum_{i=1}^{h} H_i \cdot (W_{A,i}^O + W_{B,i}^O), \tag{1}$$

where $W_{i,\text{combined}}^O = W_{A,i}^O + W_{B,i}^O$ is the combined output projection for head $i$. The blockwise structure ensures that each head's computation remains independent and can be algebraically combined after training.

**Feed-Forward Network Re-parameterization**: For the feed-forward network, progressive joining operates similarly. Each branch computes its first-layer output independently, but these values are progressively combined before the GELU activation:

$$\text{Branch A and B computes: } h_A = W_{1A}x,\ h_B = W_{1B}x$$
$$\text{Branch A and B GELU receives: } h_A{+}\lambda(t){\cdot}h_B,\ h_B{+}\lambda(t){\cdot}h_A$$

At convergence when $\lambda{=}1$, both branches apply GELU to $h_A + h_B = (W_{1A} + W_{1B})x$, enabling re-parameterization with $W_{1,\text{combined}} = W_{1A} + W_{1B}$. Since both branches produce identical GELU outputs, the second layer can be combined as $W_{2,\text{combined}} = W_{2A} + W_{2B}$. Please refer to Algorithm 1 in the Appendix for a detailed overview of our structural re-parameterization.

**Diversity Regularization Across Branches**: While our re-parameterization ultimately merges multiple branches into a single inference-time structure, we explicitly encourage representational diversity during training to avoid branch collapse and to exploit the benefits of temporary over-parameterization. To this end, we introduce a diversity regularization term that penalizes similarity between branch features. Given $n$ feature representations $f_1, \ldots, f_n$ produced by parallel branches, we compute the average pairwise squared cosine similarity as $\mathcal{L}_{\text{div}} = \frac{1}{\binom{n}{2}} \sum_{i<j} \mathbb{E}\big[\cos(f_i, f_j)^2\big]$, which is evaluated across both attention and FFN layers and added to the training objective. This loss explicitly discourages branches from learning identical representations, encouraging complementary feature subspaces that are subsequently fused through re-parameterization. Importantly, unlike mixture-of-experts (MoE) models that aim to preserve expert specialization at inference time (Rajbhandari et al., 2022; Dai et al., 2024), our objective is to leverage *temporary multi-branch diversity during optimization*, followed by deterministic structural merging. The regularizer operates only during training, encouraging each branch to explore distinct representations.

**Implementation Details**: Our framework transforms a standard $L$-layer transformer into a compact $L/n$-layer model for deployment, where $n$ is the compression factor. During training, instead of stacking $L$ sequential layers, we reorganize the network into $L/n$ parallel blocks, each containing $n$

branches. This preserves **the same total parameters and FLOPs** as the original model. Within each block, all branches process the input $X$ in parallel, and their outputs are aggregated by summation, allowing gradients to flow naturally through all branches. Moreover, since branching is accompanied by proportional depth reduction, the volume of intermediate activations remains unchanged. Consequently, *our training procedure requires essentially the same computational and memory cost as the baseline model*. After training, we perform structural re-parameterization by collapsing each parallel block into a single sequential layer via exact weight fusion, $W_{\text{combined}} = \sum_{i=1}^{n} W_i$, which incurs negligible computational cost. Our method is compatible with standard optimizations, including mixed precision, gradient accumulation, and distributed data parallelism. In Swin ViT (Liu et al., 2021), we apply re-parameterization exclusively to the third stage. While other stages are compressed and typically consist of just a single W-MSA/SW-MSA pair, the third stage is composed of multiple identical block pairs, which creates substantial room for compression. Furthermore, this stage is the most computationally intensive, contributing up to 80% of the total FLOPs. To study re-parameterization as a *capacity booster* rather than compression, we also consider a setting where inference depth remains $L$. Here, we train with $n$ branches per layer and merge them post-training, increasing optimization capacity while preserving the original inference architecture.

### 3.4 Compatibility with Token Reduction

To further push the efficiency frontier, we seamlessly integrate token-level acceleration into our re-parameterized architecture. We adopt ToMe (Bolya et al., 2022) for pure ViTs. However, standard ToMe relies on the attention Keys ($K$) of a single branch to determine token similarity, which is ill-suited for our design where feature representations are distributed across parallel branches. To address this, we propose a unified concatenation strategy for metric computation. Since the attention score in our re-parameterized block is derived from the sum of dot products across $M$ branches: $\sum_{i=1}^{M} Q^{(i)} {K^{(i)}}^T$), the true similarity between tokens is defined in the joint feature space of all branches. Mathematically, this is equivalent to the dot product of concatenated vectors: $[Q^{(1)}, \ldots, Q^{(M)}] \cdot [K^{(1)}, \ldots, K^{(M)}]^T$. Therefore, instead of averaging, we concatenate the Keys from all parallel branches along the feature dimension to construct the metric $\mathcal{M} = \frac{1}{H} \sum_{h=1}^{H} \text{Concat}\left(K_h^{(1)}, K_h^{(2)}, \ldots, K_h^{(M)}\right)$, where $K_h^{(i)}$ denotes the Key of the $h$-th head in the $i$-th branch, and the mean is taken over the $H$ attention heads. This ensures that the bipartite matching algorithm considers the complete, multi-view feature set when identifying redundant tokens. For multi-modal VLMs, we integrate our method with VisionZip (Yang et al., 2025). In dominant token selection, we use the CLS token's attention scores after re-parameterization to identify key visual tokens. For contextual token merging, similarities are computed from concatenated Keys across all branches in the same manner as ToMe, ensuring joint feature representations are considered.

## 4 Experiments

We conduct extensive experiments to validate the effectiveness of the proposed re-parameterized ViT across a diverse set of tasks. All experiments are performed on 8 NVIDIA H200 GPUs. Detailed experimental configurations are provided in Appendix A. To assess latency and throughput, we benchmark our models on three representative processors spanning a wide computational spectrum: a GPU (RTX 4080), a CPU (Intel i9-9900X, single-thread), and an ARM processor (Cortex-A72, single-thread). Latency is measured at batch size 1, and throughput at batch size 32.

Table 1: Evaluation of our re-parameterization method on extremely lightweight ViTs on ImageNet-1K. Our non-deep models (highlighted in gray) achieve substantially higher accuracy than their iso-depth baselines, while maintaining similar latency.

| Methods | #params | FLOPs | Throughput on GPU (fps) | Latency on CPU (ms) | Latency on ARM (ms) | Accuracy Top-1 (%) |
|---|---|---|---|---|---|---|
| DeiT-Ti-3 | 1.7M | 293M | 4114 | 28.4 | 289 | 38.7 |
| MAE-Ti-3† [1] | | | | | | 53.8 |
| DeiT-Ti-4 | 2.2M | 381M | 3756 | 32.2 | 361 | 54.2 |
| MAE-Ti-4† | | | | | | 61.0 |
| DeiT-Ti-6 | 3.1M | 556M | 2932 | 41.3 | 456 | 63.4 |
| MAE-Ti-6† | | | | | | 68.6 |
| DeiT-Ti-3-R | 1.9M | 322M | 3884 | 31.0 | 347 | **58.5** |
| DeiT-Ti-4-R | 2.4M | 415M | 3592 | 33.4 | 397 | **64.1** |
| DeiT-Ti-6-R | 3.3M | 595M | 2840 | 41.7 | 482 | **70.2** |
| MAE-Ti-3-R† | 1.9M | 322M | 3884 | 31.0 | 347 | **60.5** |
| MAE-Ti-4-R† | 2.4M | 415M | 3592 | 33.4 | 397 | **67.0** |
| MAE-Ti-6-R† | 3.3M | 595M | 2840 | 41.7 | 482 | **72.4** |

**Model setup and notation**: We adopt DeiT and Swin for supervised learning, and MAE for self-supervised learning, using identical model configurations in terms of patch size, embedding dimension, and number of attention heads. Except for MAE-Ti and variants, we follow the efficient design principles of (Wang et al., 2023) to improve performance by using 12 attention heads, replacing the class token with global average

[1] † denotes models trained with distillation.

pooling during fine-tuning, and applying knowledge distillation from an MAE-B teacher by transferring last-layer attention maps. However, our reproduced MAE-Ti-12† attains 76.4% Top-1 accuracy, compared to the reported 78.4%. We thus use this value as the reference for fair comparison in the following subsections. In multi-branch settings, the distillation loss is computed as the mean squared error between the teacher's attention maps and the averaged maps across branches. We adopt a unified naming scheme for clarity. We explicitly denote a vanilla model with depth L as `arch-`$L$, and the one with $l$ layers and our re-parameterization is written as `arch-`$l$`-R`. Depth-scaling variants without re-parameterization are denoted as `arch-`$l$. Wider variants with $n$ parallel branches are represented as `arch-`$l$-$n$. We follow the same notation to the VLM's vision encoder, by explicitly specifying the number of branches as `CLIP-ViT-[B/L]-R`$n$.

### 4.1 Evaluation on ImageNet-1K

Table 1 highlights the performance of our re-parameterized ultra-lightweight ViTs on the ImageNet-1K (Deng et al., 2009) classification task, with particular emphasis on latency and throughput across diverse hardware platforms. Our models demonstrate that substantial depth reduction can be achieved while yielding notable accuracy gains over iso-depth baselines. Specifically, **DeiT-Ti-6-R** and **MAE-Ti-6-R†** outperform their 6-layer baseline counterparts, achieving Top-1 accuracies of **70.2%** and **72.4%**, corresponding to improvements of **+6.8%** and **+6.7%**, respectively. These gains are obtained with only marginal overhead in inference cost, approximately $\sim$1% higher CPU latency, $\sim$3% lower GPU throughput, and $\sim$5% higher ARM latency, making the approach particularly attractive for edge and mobile deployment. Even under more aggressive depth reductions to 4 or 3 layers (e.g., MAE-Ti-4-R† and MAE-Ti-3-R†), our method continues to deliver substantial accuracy improvements (**67.0%** and **60.5%**, respectively), while incurring only slightly reduced throughput and moderately increased latency. Further scaling and generalization results are reported in Table 15 of the Appendix. Notably, compared with the 12-layer DeiT-Tiny, our DeiT-Ti-6-R achieves comparable accuracy with only a 2% drop, while providing substantially improved efficiency, delivering approximately 1.6$\times$ higher GPU throughput and 0.63$\times$ CPU latency. Also, our framework generalizes beyond standard ViT architectures to hybrid CNN-transformer models; results on EfficientFormer are provided in Appendix B.3.

Table 2: Comparison against SOTA works with comparable parameter budgets on ImageNet-1K. Iso-depth Baselines are adhered above the proposed methods. * denotes our reproduced result, which differs from the originally reported value, possibly due to differences in pre-training/fine-tuning data.

| Methods | #param. | FLOPs | Acc. (%) |
|---|---|---|---|
| *Supervised Convolutional models* | | | |
| MicroNet-M3 (Li et al., 2021b) | 2.6M | 21M | 62.5 |
| EtinyNet-M1 (Xu et al., 2022) | 3.9M | 117M | 65.5 |
| MobileNetV2 (Sandler et al., 2018) | 3.5M | 310M | 72.0 |
| FasterNet-T0 (Chen et al., 2023) | 3.9M | 340M | 71.9 |
| *Supervised Efficient convolutional ViT/pure ViT models* | | | |
| MobileViT-XS (Mehta & Rastegari, 2021) | 2.3M | 1.05G | 74.8 |
| PVTv2-B0v (Wang et al., 2022) | 3.4M | 600M | 70.5 |
| EdgeViT-XXS (Pan et al., 2022) | 4.1M | 600M | 74.4 |
| EfficientViT-B1 (Cai et al., 2023) | 9.1M | 0.52G | 79.4 |
| EfficientFormer-L1 (Li et al., 2022b) | 12.3M | 1.4G | 80.2 |
| *Supervised Reparameterization-based convolutional ViT/pure ViT models* | | | |
| CaiT-XXS-24 (Touvron et al., 2021b) | 12.0M | 2.5G | 77.6 |
| RePa-DeiT-Tiny/0.5 (Xu et al., 2025) | 4.4M | 800M | 69.4 |
| SLAB-DeiT-Ti (Guo et al., 2024) | - | 1.1G | 74.3 |
| FastViT-T12 (Vasu et al., 2023) | 6.8M | 1.4G | 79.1 |
| RePa-DeiT-Small/0.75 (Xu et al., 2025) | 12.2M | 2.5G | 77.0 |
| RePa-Swin-Tiny/0.75 (Xu et al., 2025) | 17.5M | 2.6G | 78.4 |
| RePa-Swin-Small/0.75 (Xu et al., 2025) | 22.9M | 5.1G | 81.4 |
| RePa-DeiT-Base/0.75 (Xu et al., 2025) | 51.1M | 9.9G | 81 |
| RePa-Swin-Base/0.75 (Xu et al., 2025) | 52.8M | 9G | 82.6 |
| *Supervised pure ViT baselines & our models* | | | |
| DeiT-Ti-6 | 3.1M | 556M | 63.4 |
| DeiT-Ti-6-R | 3.3M | 595M | 70.2 |
| DeiT-Ti-12 | 5.7M | 1.08G | 72.2 |
| DeiT-Ti-12-R | 5.9M | 1.2G | 74.5 |
| DeiT-S-6 | 11.3M | 2.2G | 71.2 |
| DeiT-S-6-R | 12.0M | 2.4G | 77.0 |
| Swin-Ti-8 | 21.2M | 3.0G | 75.6 |
| Swin-Ti-8-R | 22.4M | 3.3G | 77.9 |
| Swin-S-12 | 28.3M | 4.4G | 77.6 |
| Swin-S-12-R | 30.2M | 4.7G | 81.5 |
| DeiT-B-6 | 43.9M | 8.5G | 73.1 |
| DeiT-B-6-R | 47.0M | 9.1G | 80.3 |
| Swin-B-12 | 49.9M | 7.8G | 81.5 |
| Swin-B-12-R | 52.8M | 8.4G | 82.6 |
| *Self-supervised pure ViT baselines & our models* | | | |
| MAE-Ti-6† | 3.1M | 556M | 68.6 |
| MAE-Ti-6-R† | 3.3M | 595M | 72.4 |
| MAE-Ti-12† | 5.7M | 1.08G | 78.4/76.4* |
| MAE-Ti-12-R† | 5.9M | 1.2G | 78.6 |
| MAE-B-6 | 43.9M | 8.5G | 73.3 |
| MAE-B-6-R | 47.0M | 9.1G | 81.4 |
| MAE-L-12 | 153.0M | 29.9G | 81.7 |
| MAE-L-12-R | 162.6M | 32.G | 83.9 |

**Comparison with prior works**: Table 2 compares our low-depth, re-parameterized ViTs with previous state-of-the-art convolutional networks, hybrid convolutional-transformer models, and pure ViTs across a wide spectrum of model scales. Convolution-based models (e.g., MobileNetV2, FasterNet-T0) and hybrid CNN-ViT models (e.g., EdgeViT, MobileViT) tend to have lower FLOPs because convolution kernels are spatially shared through sliding-window operations. In contrast, pure ViTs rely on token-mixing via matrix multiplications or MLPs with no weight sharing, which inherently results in higher FLOPs for the same representational capacity. However, many convolutional and hybrid models achieve their efficiency through custom operators and irregular memory access patterns,

which are often difficult to optimize for GPUs and edge devices. As a result, their theoretical FLOP advantage does not always translate to practical speedup. In contrast, our approach uses standard ViT blocks without any custom modules or architectural "bells and whistles", ensuring broad compatibility and simplifying deployment. Our method achieves competitive accuracy while significantly reducing network depth. For instance, DeiT-S-6-R, with only 12.0M parameters and 2.4G FLOPs, achieves 77.0% Top-1 accuracy, closely matching RePa-DeiT-Small/0.75 (77.0% with 12.2M parameters and 2.5G FLOPs), while remaining competitive with CaiT-XXS-24 (77.6%) and EfficientFormer-L1 (80.2%), despite relying on a simpler and purely Transformer-based design. At larger model scales (e.g. DeiT-B-6-R and Swin-B-12-R), our re-parameterization method continues to consistently match the performance of RePa-ViT architectures.

### 4.2 Transfer Learning

We further assess the downstream transfer performance of proposed models on Flowers (Nilsback & Zisserman, 2008), Pets (Parkhi et al., 2012), CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100, and iNat-18 (Van Horn et al., 2018), with results reported in Table 3. DeiT-Ti-6-R and MAE-Ti-6-R† remain competitive with their 12-layer baselines, while the 4- and 3-layer variants are underperform on several datasets. For instance, MAE-Ti-3-R† achieves only 75.1% on Pets. On COCO (Lin et al., 2014) detection and segmentation with Mask-RCNN 1× (He et al., 2017) referring to configurations provided by (Li et al., 2021a), MAE-Ti-6-R† achieves 38.0 AP (det.) and 33.6 AP (seg.), close to the MAE-Ti-12† baseline (42.3 / 37.4). This demonstrates that re-parameterized backbones preserve the spatial information required for dense prediction.

Table 3: Transfer evaluation results. Top-1 accuracy is reported for classification tasks and AP is reported for object detection (det.) and instance segmentation (seg.) tasks.

| Methods | Datasets | | | | | COCO | |
|---|---|---|---|---|---|---|---|
| | Flowers | Pets | CIFAR10 | CIFAR100 | iNat18 | (det.) | (seg.) |
| DeiT-Ti-12 (Wang et al., 2023) | 96.4 | 93.1 | 96.1 | 85.8 | 63.6 | 40.4 | 35.5 |
| DeiT-Ti-3-R | 91.5 | 77.1 | 90.4 | 70.3 | 54.5 | 31.1 | 27.1 |
| DeiT-Ti-4-R | 94.1 | 79.3 | 93.1 | 75.6 | 60.7 | 35.1 | 31.4 |
| DeiT-Ti-6-R | 94.8 | 87.5 | 95.4 | 81.3 | 62.5 | 38.1 | 33.5 |
| MAE-Ti-12† (Wang et al., 2023) | 95.2 | 89.1 | 95.9 | 85.0 | 63.6 | 42.3 | 37.4 |
| MAE-Ti-3-R† | 90.2 | 75.1 | 90.4 | 68.0 | 53.2 | 29.2 | 26.7 |
| MAE-Ti-4-R† | 93.1 | 76.4 | 93.8 | 76.6 | 61.9 | 35.3 | 31.8 |
| MAE-Ti-6-R† | 94.2 | 82.8 | 95.2 | 81.8 | 62.8 | 38.0 | 33.6 |

### 4.3 Ablation Studies

**Depth vs. Branching vs. Re-parameterization**: We disentangle the contributions of depth and branching structures. Table 4 compares parallel branching without re-parameterization, pure depth scaling, and our proposed re-parameterized models. Simply decreasing depth without branches yields great performance degradation, while adding parallel branches without joining incurs large parameter and FLOPs overheads. In contrast, our re-parameterization strategy consistently achieves higher accuracy with fewer parameters and significantly reduced computation. For example, MAE-Ti-6-R† improves accuracy by +3.8% over its depth-scaled baseline while using only half the FLOPs compared to the naive multi-branch alternative. This demonstrates that re-parameterization not only enhances representational capacity but also provides a more efficient trade-off between accuracy and model complexity.

Table 4: Ablation on branch/depth scaling & re-parameterization.

| Method | #Param. | FLOPs | Top-1 | Δ |
|---|---|---|---|---|
| *Parallel branches* | | | | |
| MAE-Ti-3-4† | 5.7M | 1.08G | 68.8 | – |
| MAE-Ti-4-3† | 5.7M | 1.08G | 71.4 | – |
| MAE-Ti-6-2† | 5.7M | 1.08G | 75.4 | – |
| *Depth scaling w/o rep.* | | | | |
| MAE-Ti-3† | 1.7M | 293M | 53.8 | – |
| MAE-Ti-4† | 2.2M | 381M | 61.0 | – |
| MAE-Ti-6† | 3.1M | 556M | 68.6 | – |
| *Depth scaling w/ rep.* | | | | |
| MAE-Ti-3-R† | 1.9M | 322M | **60.5** | +6.7 |
| MAE-Ti-4-R† | 2.4M | 415M | **67.0** | +6.0 |
| MAE-Ti-6-R† | 3.3M | 595M | **72.4** | +3.8 |

**Lambda Scheduling and Diversity Regularization**: Table 5 reveals several key observations. Instant joining leads to a substantial performance drop, highlighting the importance of adopting a progressive joining schedule. Among different schedulers, linear interpolation consistently delivers the best accuracy and stability across model scales, whereas cosine, exponential, and square-root schedules achieve comparable but slightly inferior results. In addition, incorporating diversity regularization provides consistent gains on small-scale datasets such as CIFAR-10 (e.g., +0.6–1.6% for MAE-Ti-3-R†), while offering only marginal improvements on larger-scale benchmarks (e.g., ImageNet-1K and COCO), as further analyzed in Appendix F.3. More ablations of joining stragy are shown in Appendix E.3

Table 5: Ablation on joining configurations.

| Lambda | Div. | Top-1 (%) | | |
|---|---|---|---|---|
| | | Ti-6-R | Ti-4-R | Ti-3-R |
| no joining ($\lambda = 0$) | ✗ | 95.9 | 94.0 | 92.2 |
| instant joining ($\lambda = 1$) | ✗ | 91.0 | 86.8 | 83.4 |
| linear | ✓ | **95.2** | **93.8** | **90.4** |
| linear | ✗ | 94.9 | 92.9 | 89.8 |
| cosine | ✓ | 93.9 | 92.2 | 90.2 |
| cosine | ✗ | 93.9 | 91.7 | 89.7 |
| exp | ✓ | 94.1 | 91.6 | 87.8 |
| exp | ✗ | 94.3 | 91.2 | 84.4 |
| sqrt | ✓ | 94.6 | 91.7 | 88.9 |
| sqrt | ✗ | 94.5 | 91.4 | 87.8 |

Table 7: Performance of LLaVA with re-parameterized vision encoders and VisionZip. ‡ denotes ViSFT fine-tuned models.

| Method | Vision encoder | VQAv2 | | GQA | | SciQA-IMG | | TextVQA | |
|---|---|---|---|---|---|---|---|---|---|
| | | w/o VZip | w/ VZip | w/o VZip | w/ VZip | w/o VZip | w/ VZip | w/o VZip | w/ VZip |
| LLaVA-1.5 7B | CLIP-ViT-B | 74.1 | 71.2 | **60.9** | 58.4 | 66.6 | 65.4 | 50.8 | 50.2 |
| | CLIP-ViT-B‡ | 74.3 | 71.5 | 59.2 | 58.4 | 66.5 | 65.7 | 51.7 | 50.6 |
| | CLIP-ViT-B-R2‡ | **74.9** | 73.0 | 60.8 | 58.6 | **67.2** | 66.3 | **54.3** | 52.1 |
| | CLIP-ViT-L | 78.5 | 76.8 | 62.0 | 59.3 | 66.8 | 65.0 | 58.2 | 55.7 |
| | CLIP-ViT-L‡ | 78.7 | 76.6 | 62.1 | 59.3 | 66.8 | 66.0 | 58.4 | 56.1 |
| | CLIP-ViT-L-R2‡ | **78.9** | 77.1 | **63.6** | 61.3 | **67.9** | 67.4 | **60.1** | 57.8 |

Table 8: Compatibility with ToMe on ImageNet-1K, with 16 token pairs merged at each layer.

| Method | FLOPs | | Acc. Top-1 (%) | |
|---|---|---|---|---|
| | w/o ToMe | w/ ToMe | w/o ToMe | w/ ToMe |
| MAE-Ti-6-R† | 595M | 418M | 72.4 | 70.1 |
| MAE-B-6-R | 8.5G | 6.3G | 81.4 | 80.6 |
| MAE-L-6-R | 15.0G | 11.2G | 81.8 | 81.1 |
| MAE-L-12-R | 29.9G | 15.0G | 83.9 | 80.9 |

### 4.4 Re-parameterization as a Capacity Booster

Table 6 evaluates re-parameterization as a capacity booster on DeiT, Swin, and MAE models by introducing multiple branches during training while keeping the inference architecture unchanged. Across architectures and model scales, re-parameterization consistently improves Top-1 accuracy without incurring additional inference cost. Especially on smaller models, the gains are particularly pronounced. For example, MAE-Ti-12† improves by +1.7% and +2.2% when trained with two and three branches, respectively, while Swin-Ti-12 benefits from a similar trend, improving from 81.3% to 82.0%. MAE-B-12 also observes a clear gain of +1.5% over its vanilla counterpart.

Table 6: Capacity booster results.

| Method | Acc. Top-1 (%) | | |
|---|---|---|---|
| | Vanilla | #branch=2 | #branch=3 |
| DeiT-Ti-12 | 72.2 | 74.3 | **74.5** |
| DeiT-S-12 | 79.8 | **80.4** | - |
| DeiT-B-12 | 81.8 | **82.0** | - |
| Swin-Ti-12 | 81.3 | 81.9 | **82.0** |
| Swin-S-24 | 83.0 | **83.2** | - |
| Swin-B-24 | 83.5 | **83.5** | - |
| MAE-Ti-12† | 76.4* | 78.1 | **78.6** |
| MAE-B-12 | 83.6 | **84.1** | - |
| MAE-L-24 | 85.9 | **86.1** | - |

### 4.5 Evaluation on Vision Language Models (VLMs)

In VLMs, the vision encoder is crucial as it converts images into visual tokens that form the visual input to the language model. The quality of these tokens directly affects the reliability of downstream reasoning and generation. Our re-parameterization framework can work in this setting by applying it to the vision encoder of LLaVA (Liu et al., 2023) as a capacity booster. Specifically, we enhance the CLIP ViT using multi-branch vision-SFT (ViSFT) (Jiang et al., 2024), and re-parameterize the model into a single-branch vision encoder for inference during LLaVA fine-tuning (details shown in Appendix A.5). As summarized in Table 7, re-parameterized vision encoders consistently improve LLaVA performance across multiple VLM benchmarks under identical inference cost. Compared to standard single-branch baselines, CLIP-ViT-B-R2 achieves gains of +0.8 on VQAv2, +0.6 on SciQA-IMG, and +2.5 on TextVQA, while similar or larger improvements are observed for CLIP-ViT-L-R2. We further validate that re-parameterized shallow vision encoders nearly match full-depth baselines in the VLM setting using MAE-based encoders (Appendix D).

### 4.6 Token Reduction Results

Table 8 indicates our re-parameterized models tolerate aggressive token reduction ($r$=16) remarkably well. For instance, MAE-L-6-R achieves a 25% reduction in FLOPs (15.0G $\rightarrow$ 11.2G) with a negligible accuracy drop of only 0.7%. Even for the computationally heavy MAE-L-12-R, we save $\sim$50% FLOPs while maintaining competitive accuracy, demonstrating that our depth-/capacity-oriented re-parameterization provides a robust feature foundation for token pruning. For LLaVA, under the standard training-free VisionZip setting, CLIP ViT-L/14 reduces tokens from 576 to 192 and CLIP ViT-B/16 from 196 to 64. Table 7 shows that re-parameterization preserves its gains, demonstrating strong complementarity with token reduction.

## 5 Conclusions & Discussions

We presented a progressive re-parameterization strategy for ViTs that trains multiple parallel branches and gradually fuses them into a single streamlined model for deployment. By explicitly addressing the main barriers to transformer re-parameterization, including non-linear activations, LayerNorm, and MHSA concatenation, our method enables exact algebraic merging without approximation loss. This allows the model to benefit from the expressive capacity and optimization flexibility of a wider multi-branch structure during training, while deploying as an ultra-shallow ViT with substantially reduced sequential depth and no added inference-time complexity. This depth reduction is particularly attractive for *analog and photonic computing platforms*, where highly parallel matrix operations are efficient but deep sequential execution introduces repeated conversion, synchronization, memory-access, and control overhead. Photonic accelerators, in particular, can provide high-throughput matrix multiplication, yet their system-level efficiency can be limited by layer-by-layer optical–electrical processing and sequential dataflow. By collapsing ViTs to $\leq$6 layers while preserving competitive accuracy, our approach reduces these depth-dependent bottlenecks and improves compatibility with optical cores, mixed-signal arrays, and resource-constrained edge devices. Rather than primarily scaling model width, as is common across ViT families, we target *sequential depth* as a complementary efficiency axis that directly impacts latency, memory traffic, and deployment complexity.

### AI USE STATEMENT

Generative AI tools were used solely to assist with language editing, including improving grammar, clarity, and presentation of author-written text. They were not used to generate the research ideas, formulate the proposed method, design the experiments, produce or analyze experimental results, or make scientific claims. All AI-assisted edits were reviewed and revised by the authors to ensure technical correctness, consistency with the conducted experiments, and appropriate attribution of prior work. The authors take full responsibility for the final content of the manuscript.

### ETHICS STATEMENT

This work focuses on the design and evaluation of efficient Vision Transformer architectures and does not involve human subjects, personally identifiable information, or the collection of new sensitive data. The experiments use established research benchmarks and pretrained models in accordance with their intended research use. We do not anticipate direct ethical risks beyond those generally associated with the deployment of machine-learning models. The authors have made every effort to report the methodology, comparisons, and experimental results accurately and transparently.

### REPRODUCIBILITY STATEMENT

We provide the architectural design, structural re-parameterization procedure, training methodology, experimental settings, datasets, evaluation metrics, and implementation details required to reproduce the main results throughout the paper and supplementary material. The derivations describing how the training-time multi-branch structures are exactly collapsed into the inference-time architecture are also provided. Experimental comparisons are conducted using established Vision Transformer backbones and publicly available benchmarks, with the relevant training and evaluation settings reported in the corresponding experimental sections and appendix.

## REFERENCES

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Kartikeya Bhardwaj, James Ward, Caleb Tung, Dibakar Gope, Lingchuan Meng, Igor Fedorov, Alex Chalfin, Paul Whatmough, and Danny Loh. Restructurable activation networks. *arXiv preprint arXiv:2208.08562*, 2022.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Han Cai, Junyan Li, Muyan Hu, Chuang Gan, and Song Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17302–17313, 2023.

Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S-H Gary Chan. Run, don't walk: Chasing higher flops for faster neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12021–12031, 2023.

Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12270–12280, 2021a. doi: 10.1109/ICCV48922.2021.01205.

Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *CoRR*, abs/2106.04533, 2021b. URL `https://arxiv.org/abs/2106.04533`.

Xianing Chen, Qiong Cao, Yujie Zhong, Jing Zhang, Shenghua Gao, and Dacheng Tao. Dearkd: Data-efficient early knowledge distillation for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12052–12062, 2022.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. 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, 2020.

Damai Dai et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

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

Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Diverse branch block: Building a convolution as an inception-like unit. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10881–10890. IEEE, 2021a.

Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13733–13742, 2021b.

Xiaohan Ding, Yiyuan Zhang, Yixiao Ge, Sijie Zhao, Lin Song, Xiangyu Yue, and Ying Shan. Unireplknet: A universal perception large-kernel convnet for audio, video, point cloud, time-series and image recognition. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5513–5524. IEEE, 2024.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Yonggan Fu, Haichuan Yang, Jiayi Yuan, Meng Li, Cheng Wan, Raghuraman Krishnamoorthi, Vikas Chandra, and Yingyan Lin. DepthShrinker: A new compression paradigm towards boosting real-hardware efficiency of compact neural networks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 6849–6862. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/fu22c.html.

Ankit Goyal, Alexey Bochkovskiy, Jia Deng, and Vladlen Koltun. Non-deep networks. *Advances in Neural Information Processing Systems*, 35:6789–6801, 2022.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017a.

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

Jialong Guo, Xinghao Chen, Yehui Tang, and Yunhe Wang. Slab: efficient transformers with simplified linear attention and progressive re-parameterized batch normalization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

Jianyuan Guo, Kai Han, Han Wu, Yehui Tang, Xinghao Chen, Yunhe Wang, and Chang Xu. Cmt: Convolutional neural networks meet vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12175–12185, 2022.

Zhiwei Hao, Jianyuan Guo, Ding Jia, Kai Han, Yehui Tang, Chao Zhang, Han Hu, and Yunhe Wang. Learning efficient vision transformers via fine-grained manifold distillation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1–12, 2022.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Mu Hu, Junyi Feng, Jiashen Hua, Baisheng Lai, Jianqiang Huang, Xiaojin Gong, and Xiansheng Hua. Online convolutional reparameterization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 558–567. IEEE, 2022.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

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

Xiaohu Jiang, Yixiao Ge, Yuying Ge, Dachuan Shi, Chun Yuan, and Ying Shan. Supervised fine-tuning in turn improves visual foundation models. *arXiv preprint arXiv:2401.10222*, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pp. 5156–5165, 2020.

Anatol Khilo, Steven J. Spector, Matthew E. Grein, Amir H. Nejadmalayeri, Charles W. Holzwarth, Michelle Y. Sander, Marcus S. Dahlem, Michael Y. Peng, Michael W. Geis, Nicole A. DiLello, Jung U. Yoon, Ali Motamedi, Jason S. Orcutt, Jade P. Wang, Cheryl M. Sorace-Agaskar, Miloš A. Popović, Jie Sun, Gui-Rong Zhou, Hyunil Byun, Jian Chen, Judy L. Hoyt, Henry I. Smith, Rajeev J. Ram, Michael Perrott, Theodore M. Lyszczarz, Erich P. Ippen, and Franz X. Kärtner. Photonic adc: overcoming the bottleneck of electronic jitter. *Opt. Express*, 20(4):4454–4469, Feb 2012. doi: 10.1364/OE.20.004454. URL `https://opg.optica.org/oe/abstract.cfm?URI=oe-20-4-4454`.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Woosuk Kwon, Sehoon Kim, Michael W. Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35:24101–24116, 2022.

Juhyun Lee, Nikolay Chirkov, Ekaterina Ignasheva, Yury Pisarchyk, Mogan Shieh, Fabio Riccardi, Raman Sarokin, Andrei Kulik, and Matthias Grundmann. On-device neural net inference with mobile gpus. In *Efficient Deep Learning for Computer Vision, CVPR Workshops*, 2019.

Hongkang Li, Meng Wang, Sijia Liu, and Pin-Yu Chen. A theoretical understanding of shallow vision transformers: Learning, generalization, and sample complexity. *arXiv preprint arXiv:2302.06015*, 2023.

Jiashi Li, Xin Xia, Wei Li, Huixia Li, Xing Wang, Xuefeng Xiao, Rui Wang, Min Zheng, and Xin Pan. Next-vit: Next generation vision transformer for efficient deployment in realistic industrial scenarios. *arXiv preprint arXiv:2207.05501*, 2022a.

Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers. *arXiv preprint arXiv:2111.11429*, 2021a.

Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. *Advances in Neural Information Processing Systems*, 35:12934–12949, 2022b.

YiZhou Li. Mor-vit: Efficient vision transformer with mixture-of-recursions. *arXiv e-prints*, pp. arXiv–2507, 2025.

Yunsheng Li, Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Lu Yuan, Zicheng Liu, Lei Zhang, and Nuno Vasconcelos. Micronet: Improving image recognition with extremely low flops. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 458–467, 2021b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Ji Liu, Dehua Tang, Yuanxian Huang, Li Zhang, Xiaocheng Zeng, Dong Li, Mingjie Lu, Jinzhang Peng, Yu Wang, Fan Jiang, et al. A unified progressive depth pruner for cnn and vision transformer. *arXiv preprint arXiv:2401.06426*, 2024.

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*, pp. 10012–10022, 2021.

Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5(5):5, 2017.

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

Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021.

Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022.

Shubham Negi, Utkarsh Saxena, Deepika Sharma, and Kaushik Roy. Hcim: Adc-less hybrid analog-digital compute in memory accelerator for deep learning workloads. In *ASPDAC '25: Proceedings of the 30th Asia and South Pacific Design Automation Conference*, pp. 1–6, 2025.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pp. 722–729. IEEE, 2008.

Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In *European conference on computer vision*, pp. 294–311. Springer, 2022.

Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 377–386, 2021. doi: 10.1109/ICCV48922.2021.00043.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pp. 18332–18346. PMLR, 2022.

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

Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems*, pp. 12786–12797, 2021.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Utkarsh Saxena, Indranil Chakraborty, and Kaushik Roy. Towards adc-less compute-in-memory accelerators for energy efficient deep learning. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 624–627, 2022. doi: 10.23919/DATE54114.2022.9774573.

Abdelrahman Shaker, Muhammad Maaz, Hanoona Rasheed, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Swiftformer: Efficient additive attention for transformer-based real-time mobile vision applications. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17379–17390. IEEE, 2023.

Yichen Shen, Nicholas C Harris, Scott Skirlo, Mihika Prabhu, Tom Baehr-Jones, Michael Hochberg, Xin Sun, Shijie Zhao, Hugo Larochelle, Dirk Englund, et al. Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11(7):441–446, 2017.

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

Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

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*, pp. 10347–10357. PMLR, 2021a.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 32–42, 2021b.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778, 2018.

Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5785–5795, 2023.

Weier Wan et al. A compute-in-memory chip based on resistive random-access memory. *Nature*, 608:504–512, 2022. doi: 10.1038/s41586-022-04992-8.

Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Rep vit: revisiting mobile cnn from vit perspective. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15909–15920. IEEE, 2024.

Shaoru Wang, Jin Gao, Zeming Li, Xiaoqin Zhang, and Weiming Hu. A closer look at self-supervised lightweight vision transformers. In *International conference on machine learning*, pp. 35624–35641. PMLR, 2023.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 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, 2021.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational visual media*, 8(3):415–424, 2022.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22–31, 2021.

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European Conference on Computer Vision*, pp. 68–85, 2022.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

Kaixin Xu, Zhe Wang, Chunyun Chen, Xue Geng, Jie Lin, Mohamed M. Sabry Aly, Xulei Yang, Min Wu, Xiaoli Li, and Weisi Lin. Lpvit: Low-power semi-structured pruning for vision transformers. *arXiv preprint arXiv:2407.02068*, 2024.

Kunran Xu, Yishi Li, Huawei Zhang, Rui Lai, and Lin Gu. Etinynet: Extremely tiny network for tinyml. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 4628–4636, 2022.

Xuwei Xu, Yang Li, Yudong Chen, Jiajun Liu, and Sen Wang. Repavit: Scalable vision transformer acceleration via structural reparameterization on feedforward network layers. *arXiv preprint arXiv:2505.21847*, 2025.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 19792–19802, 2025.

Lu Yu and Wei Xiang. X-pruner: explainable pruning for vision transformers. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pp. 24355–24363, 2023. doi: 10.1109/CVPR52729.2023.02333.

Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=9jsZiUgkCZP`.

Zhongzhi Yu, Yonggan Fu, Jiayi Yuan, Haoran You, and Yingyan (Celine) Lin. Netbooster: Empowering tiny deep learning by standing on the shoulders of deep giants. In *Proceedings of the 60th Annual ACM/IEEE Design Automation Conference*, DAC '23, pp. 1–6. IEEE Press, 2025. ISBN 9798350323481. doi: 10.1109/DAC56929.2023.10247827. URL `https://doi.org/10.1109/DAC56929.2023.10247827`.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Seokju Yun and Youngmin Ro. Shvit: Single-head vision transformer with memory efficient macro design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5756–5767, 2024.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Tianxiao Zhang, Wenju Xu, Bo Luo, and Guanghui Wang. Depth-wise convolutions in vision transformers for efficient training on small datasets. *Neurocomputing*, 617:128998, 2025. doi: 10.1016/j.neucom.2024.128998.

Zhuofan Zong, Kunchang Li, Guanglu Song, Yali Wang, Yu Qiao, Biao Leng, and Yu Liu. Self-slimmed vision transformer. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pp. 432–448, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20082-3. doi: 10.1007/978-3-031-20083-0_26. URL `https://doi.org/10.1007/978-3-031-20083-0_26`.

# A EXPERIMENT SETTINGS

## A.1 PRE-TRAINING

For a fair comparison with prior work, we strictly adhere to the training protocols specified in the corresponding baseline papers. For supervised learning, we adopt the common configurations outlined in DeiT (Touvron et al., 2021a) and Swin (Liu et al., 2021). For MAE, we follows the original design in (He et al., 2022), including the optimizer, learning rate schedule, batch size, and data augmentation strategies. However, to better accommodate smaller-scale ViT encoders (MAE-Ti-12), we incorporate several adjustments provided by (Wang et al., 2023), aiming to improve training efficiency on lightweight models. Specifically, we employ an MAE decoder consisting of a single Transformer block with width 192, and set the masking ratio to 0.75, which we found to yield the best performance in our setting. The specific training settings are shown in Table 9, 10, 12, and 11.

Table 9: Pre-training setting for DeiTs.

| config | value |
|---|---|
| optimizer | AdamW |
| learning rate | 1e-3 |
| weight decay | 0.05 |
| batch size | 1024 |
| learning rate schedule | cosine decay (Loshchilov et al., 2017) |
| warmup epochs | 5 |
| training epochs | 300 |
| momentum coefficient | 0.9 |

Table 10: Pre-training setting for Swins.

| config | value |
|---|---|
| optimizer | AdamW |
| learning rate | 1e-3 |
| weight decay | 0.05 |
| batch size | 1024 |
| learning rate schedule | cosine decay |
| warmup epochs | 20 |
| training epochs | 300 |

Table 11: Pre-training setting for MAEs(B/L).

| config | value |
|---|---|
| optimizer | AdamW |
| base learning rate | 1.5e-4 (batch size$=256$) |
| weight decay | 0.05 |
| optimizer momentum | $\beta_1, \beta_2 = 0.9, 0.95$ |
| batch size | 4096 |
| learning rate schedule | cosine decay |
| warmup epochs | 40 |
| training epochs | 800 |

Table 12: Pre-training setting for MAE-Ti.

| config | value |
|---|---|
| optimizer | AdamW |
| base learning rate | 1.5e-4 (batch size$=256$) |
| weight decay | 0.05 |
| optimizer momentum | $\beta_1, \beta_2 = 0.9, 0.95$ |
| batch size | 1024 |
| learning rate schedule | cosine decay |
| warmup epochs | 40 |
| training epochs | 400 |
| momentum coefficient | 0.9 |
| masking ratio | 0.75 |
| teacher model | MAE-B |

## A.2 PROGRESSIVE JOINING

Since joining occurs immediately after the completion of pre-training, the training configuration largely follows that of the pre-training stage. Unless otherwise specified, we set the batch size as 1024, lambda warmup period to 10k steps, followed by an adjustment phase of 50k steps, and a linear lambda scheduler is adopted as the default across all experiments.

## A.3 EVALUATION WITH FINE-TUNING

As shown in Table 13, we adopt the fine-tuning configurations of (He et al., 2022) for MAE models, and, except for MAE-Ti, follow the improved protocol introduced in (Wang et al., 2023). The default hyperparameter settings are summarized in Table 13. We adopt the linear learning rate scaling rule, defined as $lr = \text{base } lr \times \frac{\text{batch size}}{256}$ (Goyal et al., 2017a), and apply layer-wise learning rate decay following (Bao et al., 2021). In contrast, supervised models such as DeiT and Swin do not undergo any additional fine-tuning after progressive joining.

Table 13: Fine-tuning setting for evaluation of MAEs.

| config | value |
|---|---|
| optimizer | AdamW |
| base learning rate | {1e-3, 1e-4} |
| weight decay | 0.05 |
| optimizer momentum | $\beta_1, \beta_2 = 0.9, 0.999$ |
| layer-wise $lr$ decay | {0.75, 0.85} |
| batch size | 1024 |
| learning rate schedule | cosine decay |
| warmup epochs | 5 |
| training epochs | {50, 100} |
| augmentation | RandAug (10, 0.5) (Ti) (Cubuk et al., 2020), RandAug (9, 0.5) (B/L) |
| colorjitter | 0.3(Only for Ti) |
| label smoothing | 0 (Ti), 0.1 (B/L) |
| mixup (Zhang et al., 2017) | 0.2 (Ti), 0.8 (B/L) |
| cutmix (Yun et al., 2019) | 0 (Ti), 1.0 (B/L) |
| drop path (Huang et al., 2016) | 0 (Ti), 0.1 (B/L) |

### A.4 Transfer Learning

For transfer learning experiments, we follow the standard setup on small-scale classification datasets (e.g., CIFAR-10/100, Flowers, Pets, Aircraft, Cars) and the protocol described in (Wang et al., 2023). Each dataset is trained using its recommended recipe, with the learning rate and number of epochs adjusted proportionally to the dataset size. Specifically, we sweep over three candidate learning rates $\{1e{-}2, 3e{-}2, 1e{-}1\}$ and three candidate epoch settings $\{(50, 10), (100, 20), (150, 30)\}$ (total epochs, warmup epochs) for each dataset, while applying layer-wise learning rate decay factors of $\{1.0, 0.75\}$. Fine-tuning is performed using SGD with momentum 0.9 and a batch size of 512. We adopt a cosine learning rate schedule with linear warm-up, and all images are resized to 224×224. Standard augmentations, including random resized cropping and horizontal flipping, are applied, without additional regularization. This setup ensures that performance comparisons across tasks primarily reflect the effect of re-parameterization. Finally, iNat18 is evaluated under the same fine-tuning settings described above.

For COCO object detection and instance segmentation we follow the 100-epoch fine-tuning recipe of (Li et al., 2021a), but replace the backbone with our ImageNet-1K pretrained, joined architecture. To match the baseline setting of (Wang et al., 2023), the all sides of the input images is resized to 640 pixels instead of 1024. Our implementation is built on detectron 2 (Wu et al., 2019).

### A.5 Vision–Language Model Setup

We detail the experimental setup for applying re-parameterization to vision–language models. Following LLaVA (Liu et al., 2023), we adopt a pretrained CLIP ViT as the visual backbone and perform vision-side supervised fine-tuning (ViSFT) (Jiang et al., 2024). To enable re-parameterization, we duplicate the pretrained CLIP ViT weights into multiple parallel branches and jointly fine-tune them during ViSFT.

For CLIP-ViT-B and variants, we apply full-parameter fine-tuning on the multi-branch vision encoder. For the larger CLIP-ViT-L, to reduce memory consumption and training overhead, we adopt LoRA-based fine-tuning with rank 64, while keeping the remaining parameters frozen. After ViSFT, all branches are algebraically merged into a single equivalent vision encoder via re-parameterization. The resulting single-branch vision encoder is then frozen and used for subsequent LLaVA fine-tuning.

All LLaVA experiments are conducted using the same language model and training protocol, ensuring that observed performance differences stem solely from the vision encoder.

To evaluate the model's performance across diverse capabilities, we employ a comprehensive suite of academic benchmarks. For general visual perception and reasoning, we adopt VQAv2 (Goyal et al., 2017b), which provides a balanced dataset to mitigate language biases, and GQA (Hudson & Manning, 2019), which specifically stresses compositional reasoning and scene understanding through complex open-ended questions. To assess domain-specific knowledge and zero-shot generalization,

we utilize ScienceQA (Lu et al., 2022), a multimodal benchmark covering diverse scientific topics with a multiple-choice format. Furthermore, we include TextVQA (Singh et al., 2019) to evaluate the model's ability to perform OCR-related reasoning and understand text embedded in natural scenes.

# B Full Results on ImageNet-1K

Table 15 provides the extended ImageNet-1K results, offering a comprehensive comparison across DeiT, Swin, and MAE architectures. We evaluate the trade-offs between vanilla deep models, depth-scaled baselines, and our re-parameterized variants (denoted with "-R") under strict constraints of model size, FLOPs, and hardware latency (GPU, CPU, and ARM).

## B.1 Main Results

**Results on DeiT:** For standard ViTs, reducing depth leads to significant performance degradation. As shown in Table 15, scaling DeiT-S from 12 to 6 layers results in an 8.6% drop in Top-1 accuracy (79.8% $\rightarrow$ 71.2%). Our re-parameterization effectively mitigates this issue. DeiT-S-6-R boosts accuracy by +5.8% (reaching 77.0%) compared to its iso-depth counterpart, while maintaining a throughput of 1104 fps, nearly 1.8$\times$ faster than the original 12-layer model (615 fps). Similarly, DeiT-B-6-R recovers 7.2% accuracy over the iso-depth baseline, achieving 80.3%. The latency overhead introduced by re-parameterization remains negligible; on ARM devices, DeiT-S-6-R incurs only a $\sim$6% latency increase (1384ms vs. 1306ms) in exchange for the substantial accuracy gain.

**Results on Swin Transformer:** In hierarchical architectures, our selective re-parameterization (applied to Stage 3) demonstrates strong generalization. Direct depth scaling severely impacts performance, with Swin-S-8 dropping to 75.6% from the original 83.0%. The re-parameterized variant, Swin-Ti/S-8-R, regains 2.6%, reaching 78.2%. More impressively, Swin-B-8-R achieves 82.2%, improving over the baseline Swin-B-8 (77.1%) by +5.1%. At 12 layers, Swin-B-12-R attains 82.6% accuracy, closely approaching the 83.5% of the 24-layer original Swin-B, while delivering 36% higher throughput (418 fps vs. 307 fps).

**Results on MAE:** The benefits of our approach are most pronounced in self-supervised masked autoencoders, where model capacity is critical for representation learning. Severe depth reduction causes drastic drops, e.g., MAE-B-6 falls to 73.3% ($-10.3$% vs. MAE-B-12). Our re-parameterization serves as a powerful capacity injector: MAE-B-6-R achieves a remarkable recovery of +8.1%, reaching 81.4%, and MAE-L-6-R improves by 5.2% over its iso-depth baseline (81.8% vs. 76.6%). These gains confirm that re-parameterization is particularly effective at compensating for capacity loss in large-scale self-supervised models.

**Intermediate Depth-Compression Ratios**: Our main experiments emphasize aggressive depth reduction, where the efficiency benefits are most pronounced. To determine whether structural re-parameterization also provides a smooth accuracy–efficiency trade-off at less aggressive compression ratios, we additionally evaluate intermediate depths using MAE-L. Starting from the 24-layer model, we construct 18-, 16-, and 12-layer re-parameterized variants. The results reveal a smooth accuracy–efficiency frontier rather than an abrupt degradation at intermediate depths. MAE-L-18-R retains 85.6% Top-1 accuracy, only 0.3 percentage points below the 24-layer baseline, while reducing both parameters and computation. MAE-L-16-R achieves 84.9% while reducing FLOPs from 59.7G to 42.8G, a reduction of approximately 28%. These results show that the proposed framework is not restricted to aggressive 2–4$\times$ depth compression, but can also expose finer-grained operating points between full-depth and highly compressed models.

Table 14: Intermediate depth compression of MAE-L on ImageNet-1K.

| Model | Params | FLOPs | Top-1 (%) |
|---|---|---|---|
| MAE-L-24 | 304.1M | 59.7G | 85.9 |
| MAE-L-18-R | 249.3M | 49.5G | 85.6 |
| MAE-L-16-R | 211.1M | 42.8G | 84.9 |
| MAE-L-12-R | 162.6M | 32.3G | 83.9 |

**Depth Scaling Law under Re-parameterization.** Across all architectures, our results reveal a consistent empirical pattern that we term the *depth scaling law under re-parameterization*: the effectiveness of our framework is strongly modulated by the target inference depth, exhibiting a clear phase transition around 6 layers.

Table 15: Evaluation of re-parameterized ViTs on ImageNet-1K.

| Methods | #parameters | FLOPs | Throughput on GPU (fps) | Latency on CPU (ms) | Latency on ARM (ms) | Accuracy Top-1 (%) |
|---|---|---|---|---|---|---|
| *DeiT-based models* | | | | | | |
| DeiT-Ti-12 (Wang et al., 2023) | 5.7M | 1.08G | 1723 | 66.1 | 793 | 72.2 |
| DeiT-S-12 (Wang et al., 2023) | 22M | 4.2G | 615 | 174.4 | 2521.7 | 79.8 |
| DeiT-B-12 (Wang et al., 2023) | 86.4M | 16.8G | 246 | 392.6 | 7237.3 | 81.8 |
| DeiT-S-6 | 11.3M | 2.2G | 1138 | 93.8 | 1306.5 | 71.2 |
| DeiT-B-6 | 43.9M | 8.5G | 465 | 206.6 | 3479.4 | 73.1 |
| DeiT-S-6-R | 12.0M | 2.4G | 1104 | 94.8 | 1384.9 | 77.0 |
| DeiT-B-6-R | 47.0M | 9.1G | 456 | 208.2 | 3637.2 | 80.3 |
| *Swin-based models* | | | | | | |
| Swin-Ti-12 (Liu et al., 2021) | 28.3M | 4.4G | 603 | 272.1 | 3492.0 | 81.3 |
| Swin-S-24 (Liu et al., 2021) | 49.6M | 8.5G | 437 | 446.0 | 5168.1 | 83.0 |
| Swin-B-24 (Liu et al., 2021) | 87.7M | 15.2G | 307 | 691.3 | 6966.9 | 83.5 |
| Swin-Ti/-S-8 | 21.2M | 3.0G | 710 | 273.4 | 2335.1 | 75.6 |
| Swin-S-12 | 28.3M | 4.4G | 508 | 321.1 | 3410.9 | 77.6 |
| Swin-B-8 | 37.2M | 5.3G | 645 | 363.8 | 2977.3 | 77.1 |
| Swin-B-12 | 49.9M | 7.8G | 445 | 442.5 | 4098.2 | 81.5 |
| Swin-Ti/S-8-R | 22.4M | 3.3G | 658 | 297.6 | 2479.8 | 78.2 |
| Swin-S-12-R | 30.2M | 4.7G | 481 | 342.9 | 3540.5 | 82.0 |
| Swin-B-8-R | 39.8M | 5.9G | 600 | 389.2 | 3096.3 | 82.2 |
| Swin-B-12-R | 52.8M | 8.4G | 418.3 | 469.6 | 4302.1 | 82.6 |
| *MAE-based models* | | | | | | |
| MAE-Ti-12† (Wang et al., 2023) | 5.7M | 1.08G | 1723 | 66.1 | 793 | 78.4 |
| MAE-B-12 (He et al., 2022) | 86.4M | 16.8G | 246 | 392.6 | 7237.3 | 83.6 |
| MAE-L-24 (He et al., 2022) | 304.1M | 59.7G | 103 | 616.3 | 10059.8 | 85.9 |
| MAE-B-4 | 29.7M | 5.7G | 620 | 169.2 | 3749.5 | 70.9 |
| MAE-B-6 | 43.9M | 8.5G | 465 | 206.6 | 3935.4 | 73.3 |
| MAE-L-4 | 52.2M | 10.1G | 390 | 164.2 | 2057.2 | 72.2 |
| MAE-L-6 | 77.4M | 15.0G | 292 | 221.9 | 2981.5 | 76.6 |
| MAE-L-12 | 153.0M | 29.9G | 179 | 358.1 | 5233.8 | 81.7 |
| MAE-B-4-R | 32.5M | 6.3G | 572 | 181.2 | 3902.8 | 77.4 |
| MAE-B-6-R | 47.0M | 9.1G | 447 | 214.2 | 4128.9 | 81.4 |
| MAE-L-4-R | 55.4M | 10.9G | 370 | 174.4 | 2119.7 | 78.1 |
| MAE-L-6-R | 81.9M | 16.2G | 280 | 241.4 | 3100.7 | 81.8 |
| MAE-L-12-R | 162.6M | 32.3G | 165 | 374.9 | 5453.3 | 83.9 |

When inference depth is $\geq$6 layers, re-parameterization consistently delivers compelling accuracy–efficiency trade-offs. For instance, MAE-L-6-R (16.2G FLOPs, 81.8%) closely approaches MAE-L-12 (29.9G FLOPs, 81.7%) at half the sequential depth, and MAE-B-6-R (81.4%) nearly matches the full MAE-B-12 baseline (83.6%) while running at nearly 2$\times$ the GPU throughput. In this regime, the recovered accuracy gains systematically widen as models scale larger (e.g., $>$5% improvement for 6-layer models across DeiT, Swin, and MAE families), confirming that re-parameterization effectively preserves hierarchical representation capacity at moderate depth.

However, as depth is further reduced to $\leq$4 layers, a fundamental representational barrier emerges. While re-parameterization still provides substantial recovery: MAE-B-4-R (77.4%) and MAE-L-4-R (78.1%) each improve by $\sim$6–7% over their iso-depth baselines, these 4-layer models nonetheless underperform models with smaller embedding dimensions but greater depth at comparable or lower compute budgets (e.g.,MAE-Ti-12). This observation is consistent with well-established theoretical understanding of depth in Transformers (Li et al., 2023). Unlike width (embedding dimension), depth enables *hierarchical representation refinement*, wherein successive layers progressively transform local features into high-level semantic abstractions, from edge detection in early layers, through mid-level part representations, to global relational reasoning in deeper layers. When depth is reduced to $\leq$4, this multi-step refinement process is truncated before high-order semantic representations can form, a capacity deficit that additional width fundamentally cannot compensate. Our results thus delineate a practical design boundary: structural re-parameterization enables aggressive yet principled ViT depth compression down to 6 layers, beyond which the irreducible demands of hierarchical feature formation impose a hard limit that cannot be overcome by training-time overparameterization alone.

### B.2 Comparison with Depth-Targeting Baselines

To further isolate the benefit of structural re-parameterization, we compare against several approaches that explicitly target shallow ViTs. We implement four controlled baselines under the same DeiT-Tiny training recipe and 300-epoch budget used in our main experiments. *LayerDrop* independently

Table 16: ImageNet-1K Top-1 accuracy (%) of alternative approaches for constructing shallow DeiT-Tiny models.

| Depth | Iso-depth | LayerDrop | Prog. Prune | Weight-Share | Depth-Adaptive | KD | NAS | Ours (-R) |
|---|---|---|---|---|---|---|---|---|
| 3 | 38.7 | 36.12 | 37.43 | 54.80 | 43.57 | – | – | **58.5** |
| 4 | 54.2 | 41.32 | 45.17 | 60.70 | 50.20 | 52.33 | 56.14 | **64.1** |
| 6 | 63.4 | 51.00 | 52.39 | 65.19 | 61.36 | 60.87 | 65.18 | **70.2** |

drops layers during training and retains the target number of layers at inference. *Progressive Depth Pruning* learns per-block importance gates with an $\ell_1$ sparsity penalty and progressively removes the lowest-scoring blocks. *Weight-Sharing* cycles a smaller set of unique blocks to recover the original 12-layer sequential depth, while sharing parameters across repeated blocks. *Depth-Adaptive* training attaches auxiliary classifiers at intermediate depths and statically truncates the network to the target depth for evaluation.

We additionally consider two strong alternatives for recovering capacity at fixed shallow depth. The *KD* baseline trains a conventional shallow student using a frozen DeiT-Ti-12 teacher, while the *NAS* baseline searches for a stronger architecture at the same target depth using an AutoFormer-style search space (Chen et al., 2021a). The NAS search varies embedding dimension, Q/K/V dimension, MLP ratio, and the number of attention heads while constraining FLOPs to within approximately $\pm 15\%$ of our re-parameterized model.

Across all evaluated target depths, structural re-parameterization provides the highest accuracy. The advantage becomes particularly pronounced under aggressive compression: at three layers, our model reaches 58.5% compared with 54.8% for the strongest alternative, while at four and six layers it reaches 64.1% and 70.2%, respectively. Importantly, the NAS results indicate that the gain cannot be recovered simply by searching for a stronger conventional architecture at the same depth and similar computational budget. These results support the role of training-time multi-branch capacity and progressive re-parameterization in recovering performance lost through depth reduction.

**Comparison with Existing Efficient ViTs.** We further compare DeiT-Ti-6-R with representative reported ViT compression methods at similar operating points. As shown in Table 17, our 6-layer model achieves an accuracy comparable to substantially less compressed methods while requiring only 0.595G FLOPs and 3.3M parameters. DeiT-Ti-6-R achieves 70.2% Top-1 accuracy, closely matching UPDP-P6 (70.3%) while using fewer FLOPs and parameters. More broadly, these results show that our approach provides a competitive accuracy–efficiency trade-off while directly reducing sequential network depth rather than only reducing within-layer computation.

Table 17: Comparison with representative efficient ViT methods on ImageNet-1K.

| Method | FLOPs | Params | Top-1 (%) |
|---|---|---|---|
| DeiT-Tiny | 1.30G | 5.4M | 72.2 |
| SCOP (Tang et al., 2020) | 0.80G | – | 68.9 |
| HVT (Pan et al., 2021) | 0.80G | – | 69.7 |
| $S^2$ViTE (Chen et al., 2021b) | 1.00G | 4.2M | 70.1 |
| UPDP-P6 (Liu et al., 2024) | 0.90G | 3.8M | 70.3 |
| **DeiT-Ti-6-R (ours)** | **0.595G** | **3.3M** | 70.2 |

### B.3 Extension to Hybrid CNN-Transformer Architectures

To demonstrate that our re-parameterization framework generalizes beyond standard ViT architectures, we evaluate it on EfficientFormer (Li et al., 2022b), a hybrid CNN-transformer model. We apply our method with minimal modifications to accommodate architecture-specific components: MB4D pooling and CNN+BN blocks are handled separately, while standard re-parameterization is applied to MB3D blocks. Models are trained for 300 epochs following the original schedule, with an additional 50 epochs for re-parameterization fine-tuning.

As shown in Table 18, our framework consistently improves performance across all EfficientFormer configurations. Re-parameterized models recover a large portion of the accuracy gap between iso-depth and full-depth baselines: EF-L1-8-R gains +4.6% over the iso-depth EF-L1-8 (75.9% vs. 71.3%), and EF-L3-13-R gains +3.1% over EF-L3-13 (80.5% vs. 77.2%). Furthermore, when used as a capacity booster at the original depth, EF-L1-15-R achieves +1.1% over the standard baseline (80.3% vs. 79.2%) at identical inference cost. These results confirm that our framework is not limited to standard ViT architectures and generalizes effectively to hybrid CNN-transformer models with minimal modifications.

Table 18: Evaluation on EfficientFormer variants on ImageNet-1K.. Depth denotes the number of blocks per stage.

| Model | Depth | Params | FLOPs | Top-1 (%) |
|---|---|---|---|---|
| EF-L1-8 (iso-depth) | [2,1,3,2] | 7.8M | 0.8G | 71.3 |
| EF-L1-15 (vanilla) | [3,2,6,4] | 12.3M | 1.4G | 79.2 |
| EF-L1-8-R | [2,1,3,2] | 7.9M | 0.9G | 75.9 |
| EF-L1-15-R | [3,2,6,4] | 12.5M | 1.6G | 80.3 |
| EF-L3-13 (iso-depth) | [2,2,6,3] | 17.1M | 2.1G | 77.2 |
| EF-L3-26 (vanilla) | [4,4,12,6] | 31.3M | 3.9G | 82.4 |
| EF-L3-13-R | [2,2,6,3] | 17.3M | 2.3G | 80.5 |
| EF-L3-26-R | [4,4,12,6] | 31.5M | 4.4G | 82.7 |

### B.4 Cross-Depth Comparisons of Re-Parameterized Models

A core claim of our framework is that models can be trained at full depth and re-parameterized into shallower architectures at inference, achieving competitive accuracy at reduced computational cost. Table 19 provides a consolidated view of these cross-depth comparisons across representative model families, including the re-parameterized model, its iso-depth counterpart (same depth, no re-parameterization), and the full-depth baseline.

Table 19: Cross-depth comparison of re-parameterized models against iso-depth counterparts and full-depth baselines. "Iso-depth" denotes a model of the same inference depth trained without re-parameterization.

| Re-param Model | Acc. (%) | Iso-depth Model | Acc. (%) | Full-Depth Baseline | Acc. (%) |
|---|---|---|---|---|---|
| DeiT-Ti-6-R | 70.2 | DeiT-Ti-6 | 63.4 | DeiT-Ti-12 | 72.2 |
| DeiT-S-6-R | 77.0 | DeiT-S-6 | 71.2 | DeiT-S-12 | 79.8 |
| DeiT-B-6-R | 80.3 | DeiT-B-6 | 73.1 | DeiT-B-12 | 81.8 |
| Swin-Ti/S-8-R | 78.2 | Swin-Ti/S-8 | 75.6 | Swin-Ti-12 | 81.3 |
| Swin-B-12-R | 82.6 | Swin-B-12 | 81.5 | Swin-B-24 | 83.5 |
| MAE-B-6-R | 81.4 | MAE-B-6 | 73.3 | MAE-B-12 | 83.6 |
| MAE-L-6-R | 81.8 | MAE-L-6 | 76.6 | MAE-L-24 | 85.9 |
| MAE-L-12-R | 83.9 | MAE-L-12 | 81.7 | MAE-L-24 | 85.9 |

Across all configurations, re-parameterized models consistently and substantially outperform iso-depth counterparts, recovering most of the accuracy gap to the full-depth baseline at half the sequential depth. For example, DeiT-Ti-6-R (70.2%) closes to within 2.0% of DeiT-Ti-12 (72.2%) while achieving $\sim$1.6$\times$ higher GPU throughput and 0.63$\times$ CPU latency. DeiT-B-6-R (80.3%) recovers 7.2% over its iso-depth baseline and comes within 1.5% of the full 12-layer model. In hierarchical architectures, Swin-B-12-R (82.6%) approaches the 24-layer Swin-B baseline (83.5%) within 0.9%, while delivering 36% higher GPU throughput. Most strikingly, MAE-B-6-R (81.4%) recovers +8.1% over the iso-depth MAE-B-6 (73.3%) at identical inference cost, closing over 75% of the gap to the full-depth MAE-B-12 (83.6%). Similarly, MAE-L-12-R (83.9%) nearly matches the 24-layer MAE-L baseline (85.9%) at half the sequential depth, with a gap of only 2.0%.

## C Transfer Learning Comparisons with Iso-Depth Baselines

To facilitate clearer interpretation of the transfer learning results in Table 3, we include the iso-depth baseline DeiT-Ti-6 alongside the re-parameterized DeiT-Ti-6-R and the full-depth DeiT-Ti-12 reference. Note that DeiT-Ti-12 (5.7M parameters) and DeiT-Ti-6-R (3.3M parameters) differ by approximately 1.7$\times$ in model size rather than exactly 2$\times$, due to re-parameterization overhead.

As shown in Table 20, DeiT-Ti-6-R consistently and substantially outperforms the iso-depth DeiT-Ti-6 across all seven benchmarks, recovering much of the gap to the full-depth DeiT-Ti-12 baseline. For instance, on Pets, DeiT-Ti-6-R achieves 87.5% compared to 74.8% for DeiT-Ti-6, closing most of the 18.3% gap to DeiT-Ti-12 (93.1%). Similar trends hold for detection and segmentation on COCO, confirming that the gains from re-parameterization transfer robustly across diverse downstream tasks.

Table 20: Transfer learning and downstream task performance of DeiT-Ti variants. DeiT-Ti-6-R uses half the inference depth of DeiT-Ti-12 while significantly outperforming the iso-depth DeiT-Ti-6 baseline across all benchmarks.

| Method | Flowers | Pets | CIFAR-10 | CIFAR-100 | iNat18 | COCO det. | COCO seg. |
|---|---|---|---|---|---|---|---|
| DeiT-Ti-12 | 96.4 | 93.1 | 96.1 | 85.8 | 63.6 | 40.4 | 35.5 |
| DeiT-Ti-12-R | 97.1 | 93.8 | 96.5 | 86.0 | 64.3 | 41.2 | 36.3 |
| DeiT-Ti-6 | 90.2 | 74.8 | 89.1 | 68.7 | 55.3 | 30.8 | 27.0 |
| DeiT-Ti-6-R | 94.8 | 87.5 | 95.4 | 81.3 | 62.5 | 38.1 | 33.5 |

# D Deploying Shallow Vision Encoders in VLMs

While Section 4.5 evaluates our framework as a capacity booster for CLIP-based vision encoders, the *re-parameterized shallow vision encoders* could not be directly validated on CLIP-ViT because CLIP pre-training data is proprietary and inaccessible, preventing us from applying the depth-reduction pipeline to CLIP-ViT. To address this, we conduct additional experiments using ImageNet-pretrained MAE (He et al., 2022) models as vision encoders within the LLaVA-1.5 7B pipeline, which allows full control over pre-training and enables direct application of our depth-reduction approach.

Specifically, we compare the following configurations:

- **MAE-{B,L}-$D$ (baseline)**: The full-depth MAE encoder with $D$ Transformer layers.
- **MAE-{B,L}-$D/2$**: A shallow iso-depth encoder with half the layers, trained from scratch.
- **MAE-{B,L}-$D/2$-R**: Our re-parameterized shallow encoder with $D/2$ layers at inference, obtained by applying the depth-reduction pipeline to the full-depth model.

Table 21: Performance of LLaVA-1.5 7B with MAE-based vision encoders under the "Deploying Shallow" setting. MAE-{B,L}-$D/2$-R denotes our re-parameterized shallow encoder; $\Delta$ reports the gain of the re-parameterized model over the iso-depth baseline.

| Vision Encoder | GQA | SciQA-IMG | TextVQA |
|---|---|---|---|
| MAE-B-12 (baseline) | 46.11 | 69.51 | 45.67 |
| MAE-B-6 | 44.09 | 66.45 | 43.57 |
| MAE-B-6-R | 46.02 | 69.64 | 44.78 |
| $\Delta$ (R vs. iso-depth) | **+1.93** | **+3.19** | **+1.21** |
| MAE-L-24 (baseline) | 46.67 | 72.17 | 45.98 |
| MAE-L-12 | 46.32 | 69.49 | 45.25 |
| MAE-L-12-R | 46.58 | 72.01 | 46.01 |
| $\Delta$ (R vs. iso-depth) | **+0.26** | **+2.52** | **+0.76** |

As shown in Table 21, our re-parameterized shallow encoders consistently and substantially outperform iso-depth counterparts across all benchmarks. Notably, MAE-B-6-R with only 6 layers closely matches the full-depth MAE-B-12 baseline, and MAE-L-12-R with 12 layers nearly recovers the performance of the 24-layer MAE-L-24 baseline. These results confirm that the "Deploying Shallow" thesis extends to the VLM setting: depth can be compressed at inference time via re-parameterization without sacrificing representation quality, even when the vision encoder is used as part of a larger multimodal pipeline.

# E Joining Strategy

## E.1 Effect of Joining Stage

We investigate how the stage at which the joining operation is introduced affects performance, especially for self-supervised methods which require fine-tuning. Intuitively, applying joining too early may interfere with stable representation learning, whereas applying it too late may reduce the model's ability to adapt to downstream tasks. We consider four strategies: **A**, applying joining immediately after pre-training; **B**, introducing joining only at the beginning of fine-tuning; **C**, delaying joining until the middle of fine-tuning (150th epoch); and **D**, applying joining only at the final stage. Our results in Table 22 indicate that introducing joining immediately after pre-training achieves the

Table 22: Effect of joining stage on ImageNet-1K performance.

| Stage | Top-1 | #epochs |
|---|---|---|
| After pre-train (A) | **72.4** | **260** |
| Fine-tune begin (B) | 71.5 | 265 |
| Mid fine-tune (C) | 70.1 | 295 |
| After fine-tune (D) | 68.9 | – |

most consistent gains, suggesting that re-parameterization benefits from guiding the optimization trajectory early on, while overly delaying the joining reduces its effectiveness.

### E.2 Ablation Study on the Length of Progressive Joining

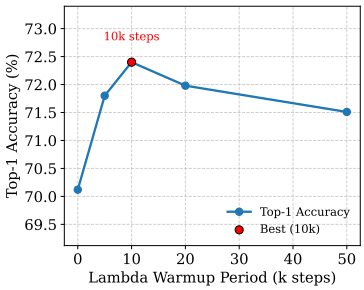



Figure 2: Effect of different Lambda Warmup Periods on ImageNet-1K MAE-Ti-6-R† fine-tuning accuracy with linear lambda scheduler. The red point indicates the best performing warmup (10k steps).

We next investigate the duration and scheduling of the progressive joining process. For clarity, we define two separate periods: (1) the **Lambda Warmup Period**, which refers to the number of steps over which the branch joining coefficient $\lambda(t)$ gradually increases from 0 to 1. We evaluate six warmup durations: 0k (instant joining), 5k, 10k, 20k, and 50k steps. (2) the **Post-Joining Adjustment Phase**, during which $\lambda$ remains at 1 and the network can further fine-tune its parameters to optimize for the fully fused representation. In our experiments, we fix this phase to 50k steps. Results on ImageNet-1K (Table 2) show that a 10k-step Lambda Warmup Period achieves the best trade-off between training efficiency and final accuracy, and we adopt it as the default in all main experiments. Instant joining (0k warmup) destabilizes optimization and yields the worst accuracy, highlighting the importance of progressive joining. Shorter warmup periods (5k) mitigate instability but remain suboptimal, while overly long warmups (20k, 50k) offer only marginal gains at the cost of unnecessarily extending the process.

### E.3 Lambda Scheduling Functions

During the progressive joining process, we employ a scheduling function $\lambda(t)$ that controls how quickly multiple branches are fused into a single branch. Here, $t \in [0, 1]$ is the normalized training progress within the Lambda Warmup Period (e.g., 10k steps). After the warmup period, the model enters the Post-Joining Adjustment Phase, where $\lambda(t)$ remains fixed at 1 and the network continues to optimize with a fully fused representation.

We explore four different joining functions:

$$\text{Linear:} \quad \lambda_{\text{linear}}(t) = t \tag{2}$$

$$\text{Cosine:} \quad \lambda_{\text{cosine}}(t) = \tfrac{1}{2}\left(1 - \cos(\pi t)\right) \tag{3}$$

$$\text{Exponential:} \quad \lambda_{\text{exp}}(t) = 1 - e^{-5t} \tag{4}$$

$$\text{Square-root:} \quad \lambda_{\text{sqrt}}(t) = \sqrt{t} \tag{5}$$

Each of these functions provides a different trade-off between early and late joining: Linear lambda scheduler increases $\lambda$ at a constant rate. Cosine scheduler starts more smoothly and accelerates later,

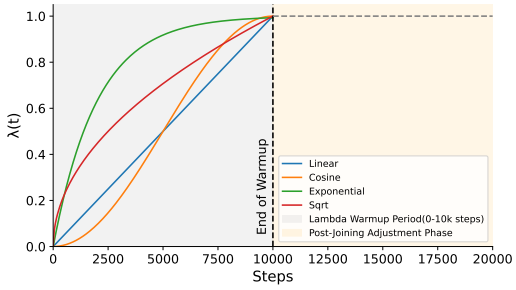



Figure 3: Visualization of different joining functions during the Lambda Warmup Period ($t \in [0, 10k]$). After the warmup (e.g., 10k steps), the Post-Joining Adjustment Phase begins with $\lambda = 1$.

which can reduce early instability. Exponential scheduler fuses branches aggressively at the beginning and quickly approaches 1. Square-root scheduler emphasizes slower growth at the beginning but accelerates faster toward the end.

Figure 3 shows the behavior of these four joining functions over the Lambda Warmup Period, followed by the Post-Joining Adjustment Phase (where $\lambda = 1$). We observe that cosine and square-root schedules provide smoother transitions compared to the linear baseline, while exponential is more aggressive.

# F Learned Representation Analysis

## F.1 Feature Space Visualization and Branch Diversity

To understand how progressive joining affects the representational capacity of our re-parameterized models, we conduct a comprehensive analysis of branch representations at different training stages. We first examine the feature learning space of our pre-trained MAE-Ti-3-4† model (i.e., the base model before joining, which we refer to as MAE-Ti-3-R†) through similarity visualization.

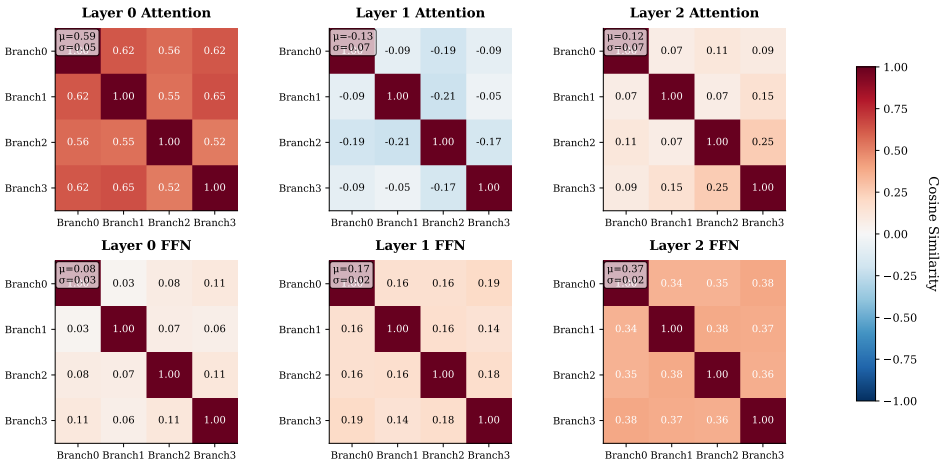



Figure 4: Feature similarity heatmap across branches in the pre-trained MAE-Ti-3-R† model before progressive joining. The visualization shows cosine similarity between feature embeddings from attention and FFN outputs of each branch.

As shown in Figure 4, we compute cosine similarity between the attention and FFN outputs of each branch to generate this heatmap. Our analysis reveals a critical insight: while Layer 0 attention modules exhibit high inter-branch similarity (average = 0.59), which can be attributed to all branches receiving identical input tokens, other modules demonstrate relatively low similarity. This finding

confirms that our base model's branches successfully learn distinct feature spaces before joining, enhancing the model's representational capacity through architectural diversity.

### F.2 Weight Similarity Analysis: Before and After Joining

Since feature similarity analysis becomes less informative after joining (due to joined input sharing by weights making high similarity inevitable), we employ an alternative approach: measuring similarity between branch weights themselves.

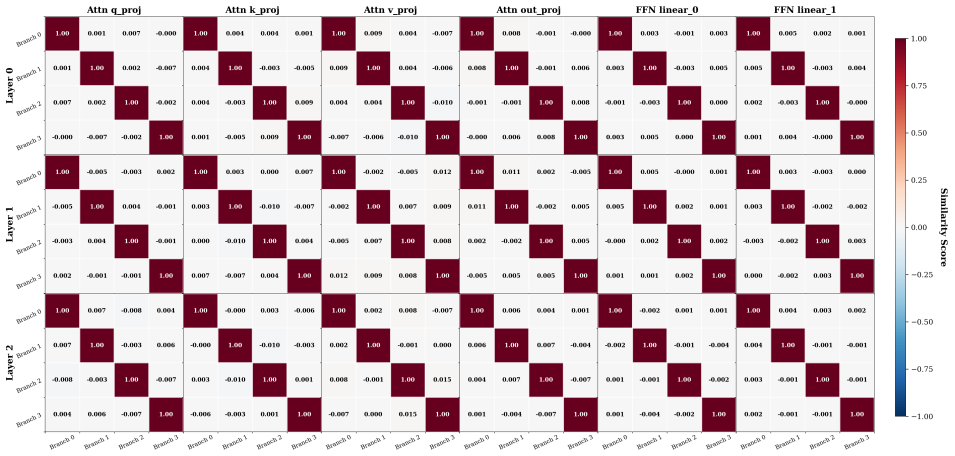



Figure 5: Weight similarity matrices across branches before progressive joining. This analysis corroborates the feature diversity observed in Figure 4, confirming that branches maintain distinct parameter spaces in MAE-Ti-3-4†.

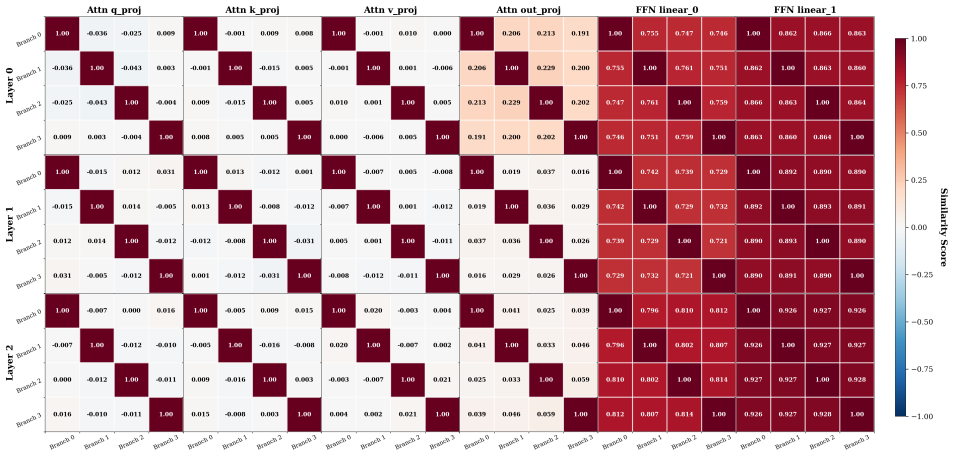



Figure 6: Weight similarity matrices after progressive joining (after Lambda Warmup Period of 10k steps and Post-Joining Adjustment Phase of 50k steps). Results show concerning overfitting patterns without diversity regularization, particularly in FFN modules.

Figure 5 corroborates our observations from Figure 4, providing additional evidence that the base 3-layer, 4×-wide ViT model with branches effectively enhances representational capacity through maintained diversity.

### F.3 Post-Joining Overfitting Challenges

Figure 6 reveals a critical challenge in our progressive joining approach. After completing the Lambda Warmup Period (10k steps) and Post-Joining Adjustment Phase (50k steps) without diversity

regularization, we observe alarmingly high similarity in FFN weights (0.725-0.928), suggesting potential overfitting behavior. Interestingly, our experiments reveal dataset-dependent recovery patterns.

**Large Dataset Recovery**: When continued fine-tuning on ImageNet-1K for 300 epochs without diversity regularization, FFN weight similarities naturally decrease to very low values ($< 0.02$), indicating that large-scale data provides sufficient regularization through its inherent diversity.

**Small Dataset Vulnerability**: On smaller datasets (e.g. CIFAR10 and CIFAR100) with transfer learning, FFN weight similarities remain critically high (0.98-1.0) when continue conduct fine-tuning, suggesting severe overfitting where branches collapse to nearly identical representations. This phenomenon likely occurs because limited data cannot provide the natural regularization effect observed with larger datasets.

## G Training Cost and Memory Analysis

Our re-parameterization framework incurs higher training cost than an iso-depth shallow baseline, but lower than the full-depth model. This is a deliberate design choice: additional compute is invested during training to produce a compact, shallow model for deployment, benefited over the vastly larger number of inference calls in real-world pipelines.

Table 23 reports per-step training time and peak GPU memory during training at batch size 128 on a single GPU, and Table 24 reports peak inference memory under identical settings on a single NVIDIA H200 GPU (image size 224×224, batch size 256, FP32, no AMP).

Table 23: Per-step training time (ms) and peak GPU memory (GB) at batch size 128 on a single NVIDIA H200.

| Model | Fwd (ms) | Bwd (ms) | Total (ms) | Memory (GB) |
|---|---|---|---|---|
| DeiT-Ti-6 | 32.76 | 53.33 | 86.09 | 5.29 |
| DeiT-Ti-6-R | 56.62 | 100.71 | 157.33 | 8.66 |
| DeiT-Ti-12 | 68.14 | 120.73 | 188.87 | 9.86 |
| DeiT-Ti-12-R | 120.65 | 172.90 | 293.55 | 15.14 |

Table 24: Peak GPU memory consumption (GB) during inference on a single NVIDIA H200.

| Model | params(M) | Peak Memory (GB) | Relative |
|---|---|---|---|
| DeiT-Ti-3-R† | 1.9 | 1.29 | 0.99× |
| DeiT-Ti-4-R† | 2.4 | 1.29 | 0.99× |
| DeiT-Ti-6† | 3.1 | 1.29 | 0.99× |
| DeiT-Ti-6-R† | 3.3 | 1.29 | 0.99× |
| DeiT-Ti-12† | 5.7 | 1.30 | 1.00× |
| DeiT-Ti-12-R† | 5.9 | 1.30 | 1.00× |

The extra memory cost is concentrated in the training phase. For DeiT-Ti-6-R, training requires 83% of the per-step cost of DeiT-Ti-12 (157.33 ms vs. 188.87 ms) while achieving competitive accuracy (70.2% vs. 72.2%). By contrast, inference memory is essentially unchanged: our progressive re-parameterization folds all branches prior to deployment, yielding negligible inference overhead (≈0.99×). The resulting deployment savings are substantial: DeiT-Ti-6-R achieves $1.6\times$ higher GPU throughput and $0.63\times$ CPU latency compared to DeiT-Ti-12.

## H Attention Map Visualization

We have provided attention map visualizations (Fig. 7) comparing three models: MAE-Ti-12 (baseline), MAE-Ti-6, and MAE-Ti-6-R. Since MAE-Ti models do not use a CLS token, we compute attention maps by averaging incoming attention across all heads and all query patches (i.e., how much each patch is attended to by all other patches), providing a spatial summary of each model's feature extraction behavior. The key differences between MAE-Ti-6 and MAE-Ti-6-R are highlighted with red bounding boxes. We observe that MAE-Ti-6-R produces attention patterns that are noticeably more focused on semantically relevant regions, closely resembling the 12-layer baseline, whereas MAE-Ti-6 tends to produce more diffuse and less discriminative attention maps. This confirms that

our re-parameterization effectively recovers the representational quality lost from depth reduction, enabling shallow models to attend to task-relevant features in a manner comparable to their deeper counterparts.

# I Algorithmic Description of Progressive Re-parameterization

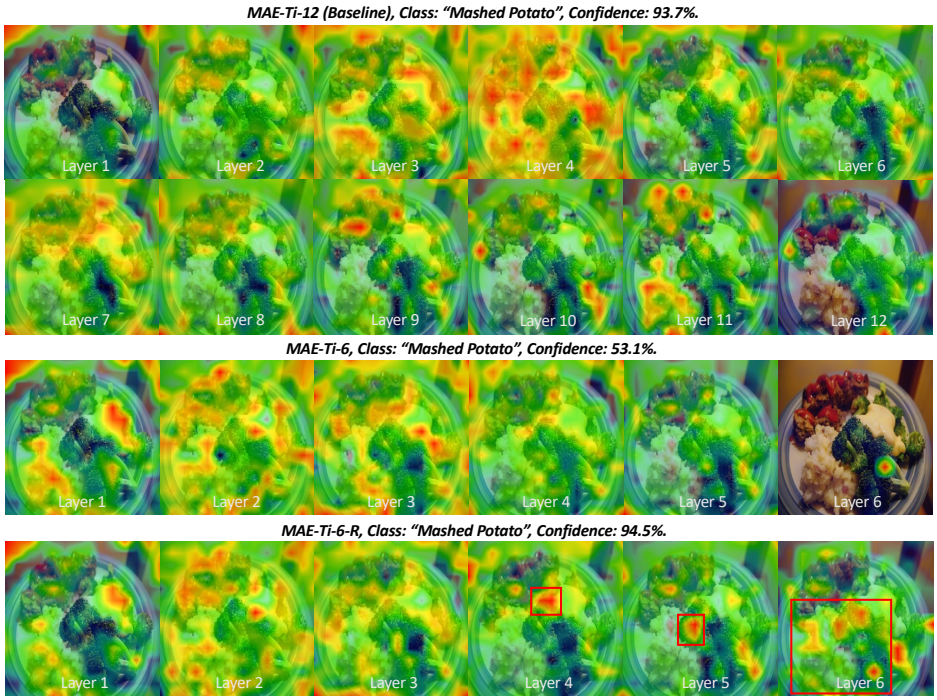
The core training algorithm for our proposed progressive structural re-parameterization is outlined in Algorithm 1. At a high level, the algorithm operates as follows:

Starting from the input, parallel transformer branches are initialized, and the MHSA-preceding LayerNorm is applied once before branch splitting so that all branches receive identically normalized inputs. The FFN-preceding LayerNorm is removed by fusing its affine parameters into the first FFN linear layer. Within each layer of each branch, attention is computed in a blockwise manner, replacing the traditional concatenation of heads with independent projection and summation per head, as described earlier.

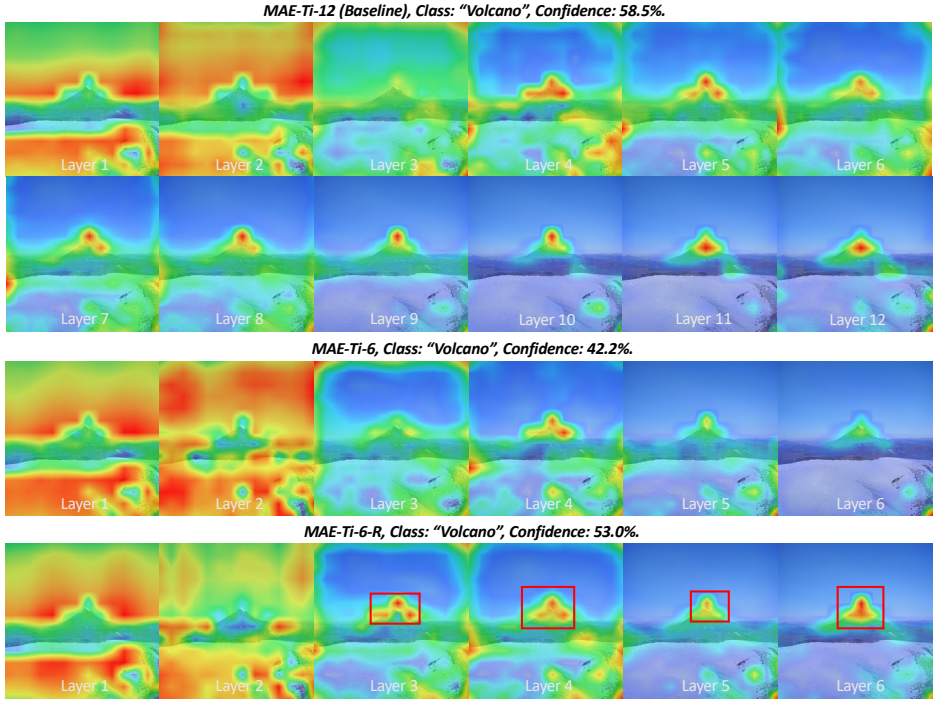
The main novelty arises in the attention computation: every head in every branch independently computes its own $Q$, $K$, and $V$ projections, then combines $Q$ and $K$ weights into a single matrix $W_{b,h}$ for that branch and head. This enables computation of attention scores as $XW_{b,h}X^T$, which are suitable for algebraic combination across branches. Progressive joining is achieved by mixing scores from both branches according to a schedule parameter $\lambda(t)$, with proper scaling to maintain numerically stable variance. These scores are softmaxed and the resulting attention is applied to values $V_{b,h}$, with output projections performed in a blockwise (head-wise) fashion.

The process also applies to the feedforward (FFN) sublayers, where branch merging occurs at the non-linear activation. Over the course of training, $\lambda(t)$ is smoothly increased so that, at convergence, all branches are merged and the final step algebraically collapses the model into a single-branch network by summing branch weights.

This procedure guarantees identical sequential and parallel computation at deployment while enabling richer representations during training. The pseudocode thus expresses: (1) blockwise attention and output projection, (2) branch-wise attention score mixing, (3) correct numerical scaling for score joining, and (4) final algebraic fusion for efficient inference. The entire routine is efficient and hardware-friendly, requiring no special kernels or additional barriers for parallel deployment.




(a) "Mashed Potato" image




(b) "Volcano" image

Figure 7: Visualization of attention maps for two images from three models: MAE-Ti-12 (baseline), MAE-Ti-6, and MAE-Ti-6-R. (a) shows the "Mashed Potato" image, and (b) shows the "Volcano" image. The primary differences in attended regions between MAE-Ti-6 and MAE-Ti-6-R are highlighted with red bounding boxes.

**Algorithm 1:** Progressive Structural Re-parameterization for Non-Deep Vision Transformers

**Input:** Input sequence $X \in \mathbb{R}^{N \times d}$, number of layers $L$, number of branches $B$, progressive parameter $\lambda(t)$

**Output:** Single-branch collapsed Vision Transformer for efficient inference

Initialize parallel transformer branches $\{\mathcal{B}_1, \ldots, \mathcal{B}_B\}$;

**for** *each training step* $t$ **do**

  **for** *each layer* $\ell = 1$ *to* $L$ **do**

    **MHSA pre-LN (shared):** Apply $\mathrm{LN}_{\mathrm{mhsa}}$ once before branching;

    **FFN LN fusion:** Absorb affine parameters $(\gamma_{\mathrm{ffn}}, \beta_{\mathrm{ffn}})$ directly into the first FFN linear layer;

    **for** *each branch* $b = 1$ *to* $B$ **do**

      **Multi-Head Attention**;

      **for** *each head* $h = 1$ *to* $H$ **do**

        Compute $Q_{b,h} = XW^Q_{b,h}$, $K_{b,h} = XW^K_{b,h}$, $V_{b,h} = XW^V_{b,h}$;

        Compute $W_{b,h} = W^Q_{b,h}(W^K_{b,h})^T$;

        Compute attention scores: $A_{b,h} = XW_{b,h}X^T$;

        Progressive joining:

$$A^{\text{join}}_{b,h} = X\left(W_{b,h} + \lambda(t)W_{b',h}\right)X^T, \quad b' \neq b$$

        Scale scores:

$$A^{\text{join}}_{b,h} \leftarrow \frac{A^{\text{join}}_{b,h}}{(1+\lambda^2(t))\sqrt{d_k}}$$

        Apply softmax: $S_{b,h} = \mathrm{softmax}(A^{\text{join}}_{b,h})$;

        Compute head output: $H_{b,h} = S_{b,h}V_{b,h}$;

      Blockwise output projection:

$$O_b = \sum_{h=1}^{H} H_{b,h}W^O_{b,h}$$

    **Feedforward Network:** Apply FFN with absorbed LN parameters;

    **Progressive joining:** Merge branch outputs at non-linear activations (GELU, softmax) using $\lambda(t) = t$, where $t$ is linearly increased from 0 to $T$ during training;

**Collapse branches:** At inference, set $\lambda = 1$ and combine all branch weights:

$$W^{\text{final}} = \sum_b W_b$$

Deploy single-branch, non-deep transformer for efficient inference;