EA-ViT: Efficient Adaptation for Elastic Vision Transformer

Chen Zhu[1,2*]   Wangbo Zhao[1†]   Huiwen Zhang[2]
Yuhao Zhou[1]   Weidong Tang[1,2]   Shuo Wang[3]   Zhihang Yuan[3]
Yuzhang Shang[4]   Xiaojiang Peng[5]   Kai Wang[1]   Dawei Yang[3†]
[1]National University of Singapore   [2]Xidian University   [3]Houmo AI
[4]UCF   [5]Shenzhen Technology University

Abstract

Vision Transformers (ViTs) have emerged as a foundational model in computer vision, excelling in generalization and adaptation to downstream tasks. However, deploying ViTs to support diverse resource constraints typically requires retraining multiple, size-specific ViTs, which is both time-consuming and energy-intensive. To address this issue, we propose an efficient ViT adaptation framework that enables a single adaptation process to generate multiple models of varying sizes for deployment on platforms with various resource constraints. Our approach comprises two stages. In the first stage, we enhance a pre-trained ViT with a nested elastic architecture that enables structural flexibility across MLP expansion ratio, number of attention heads, embedding dimension, and network depth. To preserve pre-trained knowledge and ensure stable adaptation, we adopt a curriculum-based training strategy that progressively increases elasticity. In the second stage, we design a lightweight router to select submodels according to computational budgets and downstream task demands. Initialized with Pareto-optimal configurations derived via a customized NSGA-II algorithm, the router is then jointly optimized with the backbone. Extensive experiments on multiple benchmarks demonstrate the effectiveness and versatility of EA-ViT. The code is available at https://github.com/zcxcf/EA-ViT.

1. Introduction

Vision Transformers (ViTs) [10, 13, 24, 47, 62] have significantly advanced the field of computer vision, delivering state-of-the-art results across a wide range of tasks [4, 29, 45, 60]. Pre-trained on large-scale datasets, ViTs exhibit strong generalization ability, enabling effective adaptation to diverse downstream applications. Despite these advantages, deploy-
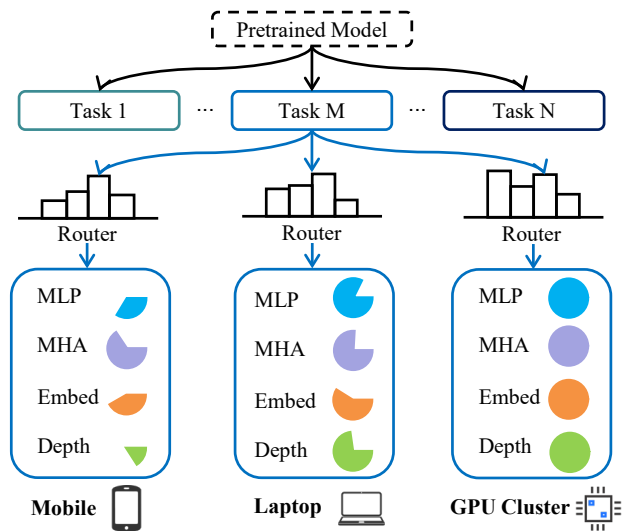

Figure 1. **Overview of the proposed EA-ViT framework.** Starting from a pre-trained ViT, EA-ViT generates task-specific elastic submodels through a router module. For each downstream task, the router selects a suitable submodel configuration—varying in MLP expansion ratio, number of attention heads, embedding dimension, and network depth. This ensures compatibility with the computational constraints of various target platforms, such as mobile devices, laptops, and GPU clusters.

ing ViTs in real-world scenarios presents notable challenges, particularly in selecting the appropriate model size to suit diverse target platforms, ranging from resource-constrained mobile devices to high-performance GPU clusters.

A common approach to tackle this problem is to apply model pruning methods [5, 14, 18, 23, 30], which reduce the model size to an appropriate level, followed by fine-tuning to restore performance for each downstream task. However, this strategy is inflexible and presents two key challenges: *i)* Training separate models for each target device significantly increases computational cost and time requirement. *ii)* Managing and storing multiple model variants complicates deployment and version control.

---

*Work done during his undergraduate internship at NUS.
†Corresponding author.

To support flexible development across different devices, recent works [2, 9, 12, 20] explore incorporating multiple submodels within a *single* network. These methods allows for the selection of a model with an appropriate size, eliminating the need for additional training. While promising, these methods still face notable limitations: *i)* These methods usually introduce elastic structures during the pre-training phase of ViTs, which requires additional fine-tuning for each submodel when transferred to downstream tasks. *ii)* More seriously, they typically support only a limited range of submodel sizes and rely on exhaustive search to find models that meet the requirements. As a results, *efficiently deploying pre-trained ViTs across diverse devices and downstream tasks remains a pressing and unresolved challenge*.

Motivated by these challenges, we propose an **E**fficient **A**daptation framework for **Vi**sion **T**ransformers (EA-ViT). As illustrated in Figure 1, our method applies elastic adaptation to a pre-trained ViT, producing a task-specific elastic variant tailored for downstream tasks. To support efficient deployment, we introduce a lightweight router that dynamically selects the optimal submodel configuration to meet the resource constraints of heterogeneous target platforms.

Our method is composed of two stages. In the first stage, we initially equip a pre-trained ViT with a nested elastic architecture. This design introduces structural flexibility along four key dimensions: MLP expansion ratio, number of attention heads, embedding dimension, and network depth. To effectively adapt the model while preserving its pre-trained knowledge, we employ a curriculum-based adaptation strategy that progressively incorporates smaller submodels, ensuring stable and efficient optimization.

In the second stage, we design a lightweight router to determine the submodel configuration across four dimensions based on task characteristics and computational constraints. To guide the router toward promising regions of the configuration space, we employ a customized NSGA-II algorithm to identify a set of candidate submodels that balance accuracy and computational cost. These configurations are used to initialize the router, which is then jointly optimized with the backbone to enable fine-grained exploration around the initial candidates. This method helps us to approach Pareto-optimal submodels and leverage limited training resources effectively to identify better-performing configurations.

Overall, EA-ViT introduces a unified adaptation framework that achieves an effective balance between accuracy and resource efficiency across various downstream tasks. We demonstrate the effectiveness of our approach on a wide range of benchmarks, including image classification, segmentation, and real-world datasets from domains such as medical imaging and remote sensing. Our main contributions are as follows:

- We introduce the first elastic adaptation framework for ViTs, enabling elasticity during the adaptation stage and

Table 1. **Comparison between EA-ViT and previous approaches.** EA-ViT supports a broader range of elastic dimensions, resulting in a significantly larger submodel space. Taking ViT-Base as an example, our method generates up to 10^{12} times more submodels than the most flexible existing method.

Method	MLP	MHA	Embed	Depth	Submodels
DynaBERT [20]	✓	✓	✗	✓	12
MatFormer [9]	✓	✗	✗	✗	10^7
HydraViT [12]	✗	✓	✓	✗	10
Flextron [3]	✓	✓	✗	✗	10^{14}
EA-ViT (ours)	✓	✓	✓	✓	10^{26}

significantly reducing both training and storage costs
- We propose a multi-dimensional elastic ViT design along with a lightweight router that selects optimal submodels based on device constraints and downstream task demands supported by an efficient training pipeline.
- Extensive experiments on multiple datasets demonstrate that our method provides broader submodel size coverage and consistently achieves higher accuracy under varying constraints compared to existing elastic approaches.

2. Related Work

Network Architecture. Determining the optimal network architecture has long been a central goal in neural network research, irrespective of whether the networks are convolution-based (e.g., CNNs [17]) or attention-based (e.g., Transformers [10]). Early efforts relied on manual design guided by empirical heuristics [21, 27]. Subsequently, Neural Architecture Search (NAS) [31, 42, 43] automated this design process by systematically exploring large architectural spaces, though at substantial computational cost. However, NAS-based approaches require re-searching and retraining models to satisfy the specific resource constraints whenever deploying to new platforms, limiting their practical scalability. To enhance deployment flexibility, recent approaches such as Once-for-All (OFA) [2] and Slimmable Networks [54] embed multiple submodels within a single elastic network, but remain mostly confined to convolutional architectures.

Elastic Transformer. Recently, elastic deployment strategies for Transformers [48] have gained increasing attention. DynaBERT [20] leverages knowledge distillation to train width- and depth-adaptive models. MatFormer [9] adopts a Matryoshka-style design to control MLP expansion ratios, generating exponentially many submodels. HydraViT [12] introduces elasticity in embedding size and attention heads, but only supports a limited number of submodels. Flextron [3] employs a router-based mechanism to build elastic MLP and MHA modules, yet relies on a surrogate model for training and focuses primarily on large language models. In

contrast, as shown in Table 1, our work targets Vision Transformers (ViTs) [10] and introduces broader elasticity across four architectural dimensions: MLP expansion ratio, number of attention heads, embedding dimension, and network depth. More importantly, none of the aforementioned methods consider adapting a large, pre-trained ViT into an elastic structure during downstream task adaptation—a challenging yet crucial direction this work seeks to address.

Dynamic Network. Previous studies have explored dynamic adjustments of the computational graph conditioned on input samples to balance performance and efficiency. Typical approaches include dynamically varying the network depth [15, 46] or width [16, 19, 28, 58, 59], or selecting specialized subnetworks, such as mixture-of-experts (MoE) [36, 55, 63] structures. In the context of Transformers, recent works have also introduced dynamic mechanisms along the token dimension [37, 56, 57]. For example, [41] uses top-k token routing to dynamically allocate compute across sequence positions under a fixed budget and [22] leverages nested Mixture-of-Experts to prioritize tokens and adapt computation under varying budgets. However, these methods mainly adapt network structures according to input-dependent criteria. In contrast, our method explicitly selects submodel configurations based on computational constraints and downstream task characteristics.

3. Method

As illustrated in Figure 2, the training procedure of EA-ViT comprises two stages. In Stage 1, we first construct a Multi-Dimensional Elastic Architecture (Section 3.1) and then apply a Curriculum-based Elastic Adaptation strategy (Section 3.2) for preliminary training of the elastic structure. In Stage 2, we employ a customized NSGA-II algorithm to initialize the router (Section 3.3) and subsequently train the router jointly with the backbone network (Section 3.4).

3.1. Multi-Dimensional Elastic Architecture

Each block in a ViT consists of two core components: a Multi-Head Attention (MHA) module and a Multi-Layer Perceptron (MLP) module, both connected via residual pathways and followed by Layer Normalization. The primary architectural factors affecting model capacity and computational cost are the MLP expansion ratio, number of attention heads, embedding dimension, and network depth.

Leveraging these dimensions, we construct an elastic ViT architecture that supports flexible configuration along all four axes. The embedding dimension is fixed across all layers, while the number of attention heads and MLP expansion ratios can vary per layer. The depth dimension controls whether we apply residual connections or skip blocks via identity mappings. These dimensions together define a large
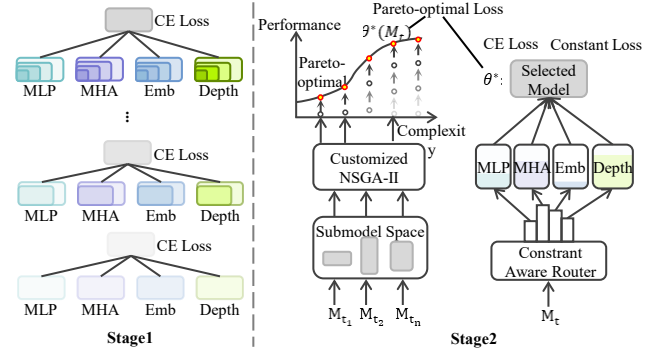

Figure 2. **Overall pipeline of our method.** In Stage 1, we construct a Multi-Dimensional Elastic Architecture and introduce elasticity into a pre-trained ViT through Curriculum-based Elastic Adaptation. In Stage 2, Pareto-optimal Submodel Search is employed to identify promising submodel configurations used to initialize the router. Then, the router and backbone are jointly optimized to further enhance submodel performance.

architectural search space, denoted as $\theta \in \Theta$, where

$$\theta = \left(\{R^{(l)}\}_{l=1}^{L}, \{H^{(l)}\}_{l=1}^{L}, E, \{D_{\text{MLP}}^{(l)}, D_{\text{MHA}}^{(l)}\}_{l=1}^{L} \right). \quad (1)$$

Here, the components are defined as follows:
- L denotes the total number of network **L**ayers.
- $R^{(l)}$ indicates the expansion **R**atio of the hidden dimension in the MLP at the l-th layer.
- $H^{(l)}$ represents the number of active attention **H**eads at the l-th layer.
- E denotes the unified **E**mbedding dimension shared across all layers.
- $D_{\text{MLP}}^{(l)}$ and $D_{\text{MHA}}^{(l)}$ are binary indicators denoting whether the MLP and MHA blocks are activated at the l-th layer, respectively, thereby adjusting the network **D**epth.

To promote parameter sharing across submodels, we adopt a nested elastic structure. Embedding dimensions, MLP hidden units, and attention heads are ranked by estimated importance, allowing more critical components to be shared across a broader range of submodels. Additional details on the ranking methodology are provided in the supplementary material A.1.

3.2. Curriculum Elastic Adaptation

Training numerous submodels simultaneously can lead to parameter interference due to conflicting gradient signals. To mitigate this, we employ a phased curriculum learning strategy that gradually expands model elasticity. Specifically, training starts with minimal elasticity — using the original pre-trained model — and progressively introduces a broader range of submodels, including smaller variants. This controlled expansion follows curriculum learning principles, moving from simpler to more complex configurations.

Algorithm 1: Curriculum Elastic Adaptation

Input: Pre-trained ViT backbone with L blocks : $\mathcal{M}_0$;   Total training steps : T;   Expansion step set : $\mathcal{S}_{\exp} \subset \{1, \ldots, T\}$;
       Initial upper bounds: $R_{\max}$, $H_{\max}$, $E_{\max}$, $n_{\max}$;   Per-expansion changes: Δ_R, Δ_H, Δ_E, Δ_n;
  Output: Adapted elastic ViT $\mathcal{M}_{\text{elastic}}$

---
1   **// Step 0: Initialize sampling ranges (start with largest model)**
2   $R_{\min} \leftarrow R_{\max}$; $H_{\min} \leftarrow H_{\max}$; $E_{\min} \leftarrow E_{\max}$; $n_{\max} \leftarrow 0$;

3   **for** $t \leftarrow 1$ **to** T **do**
4   **// Step 1: Expand elasticity at scheduled steps**
5   **if** $t \in \mathcal{S}_{\exp}$ **then**
6         $R_{\min} \leftarrow R_{\min} - \Delta_R$; $H_{\min} \leftarrow H_{\min} - \Delta_H$; $E_{\min} \leftarrow E_{\min} - \Delta_E$; $n_{\max} \leftarrow n_{\max} + \Delta_n$
7   **end**

8   **// Step 2: Sample submodel hyperparameters**
9      $R \sim \mathcal{U}(R_{\min}, R_{\max})$; $H \sim \mathcal{U}(H_{\min}, H_{\max})$; $E \sim \mathcal{U}(E_{\min}, E_{\max})$; $s \sim \mathcal{U}\{0, \ldots, n_{\max}\}$
10   Select $S \subseteq \{1, \ldots, L\}$ with $|S| = s$; for $l = 1$ to L: $D^{(l)} \leftarrow 0$ if $l \in S$, else 1;;

11   **// Step 3: Build and train sampled submodel**
12      $\mathcal{M}_t \leftarrow \text{BUILDSUBMODEL}(\mathcal{M}_{t-1}, R, H, E, \{D^{(l)}\})$; $loss \leftarrow \mathcal{L}(\mathcal{M}_t(\text{data}))$; Backpropagate $\nabla_\theta loss$ and update weights;
13   **end**
14   **return** $\mathcal{M}_{\text{elastic}} \leftarrow \mathcal{M}_T$

At the initial stage, we train only the largest submodel. As training progresses, we incrementally expand the sampling ranges for the architectural parameters defined in Equation 1. Specifically, the MLP expansion ratio R, the number of attention heads H, and the embedding dimension E are each sampled from uniform distributions with gradually decreasing lower bounds as a function of the training step:

$$R \sim \mathcal{U}(R_{\min}^{(t)}, R_{\max}),\ H \sim \mathcal{U}(H_{\min}^{(t)}, H_{\max}),\ E \sim \mathcal{U}(E_{\min}^{(t)}, E_{\max}), \tag{2}$$

where the lower bounds $R_{\min}^{(t)}, H_{\min}^{(t)}, E_{\min}^{(t)}$ are updated at scheduled training steps to progressively enlarge the submodel space. For the depth parameter D, we adopt a stochastic sampling strategy to control the number of skipped layers. Initially, all layers are active, i.e., $D^{(l)} = 1$ for all $l = 1, \ldots, L$. As training proceeds, we gradually increase the number of skipped layers. Specifically, for a given number n, we randomly select a subset $S \subseteq \{1, \ldots, L\}$ with $|S| = n$, and set $D^{(l)} = 0$ for all $l \in S$, indicating the corresponding layers are skipped. The full procedure is presented in the pseudocode shown in Algorithm 1.

3.3. Pareto-Optimal Submodel Search

To accelerate router convergence within the exponentially large submodel space, we first identify a set of promising candidate submodels for initialization. We leverage the NSGA-II evolutionary algorithm [7] as it is inherently well-suited for this task: its crossover operation is akin to the structured recombination of architectural components across multiple dimensions, while its mutation process promotes diversity and prevents premature convergence, enhancing the robustness of submodel discovery.

However, directly applying standard NSGA-II can introduce several challenges in our setting, including redundancy among selected individuals and insufficient coverage across different model complexity levels. To mitigate these issues, we introduce a partitioned selection strategy combined with an iterative crowding-distance-based selection mechanism. Specifically, we partition the normalized complexity space and perform independent selection within each partition, while enforcing a minimum complexity gap between selected individuals. This approach significantly improves population diversity and ensures broad and balanced coverage of the submodel search space. Further implementation details are provided in the supplementary material A.2.

3.4. Constraint-Aware Router

Deploying ViTs in real-world settings requires selecting submodel configurations that align with the resource constraints of the target device. We introduce a constraint-aware router that predicts the optimal submodel configuration given a specific resource budget, such as the number of Multiply-Accumulate operations (MACs). While MACs are the primary constraint considered in this work, the framework is easily extensible to other metrics such as parameter count.

Construction. We implement the router as a lightweight two-layer MLP that maps a normalized target constraint $M_t \in [0, 1]$ to a corresponding submodel configuration θ. Given the normalized MACs budget as input, the router outputs the architectural parameters defined in Equation 1:

$$\theta = \left(\{R^{(l)}\}_{l=1}^L, \{H^{(l)}\}_{l=1}^L, E, \{D_{\text{MLP}}^{(l)}, D_{\text{MHA}}^{(l)}\}_{l=1}^L \right) = \text{Router}(M_t). \tag{3}$$
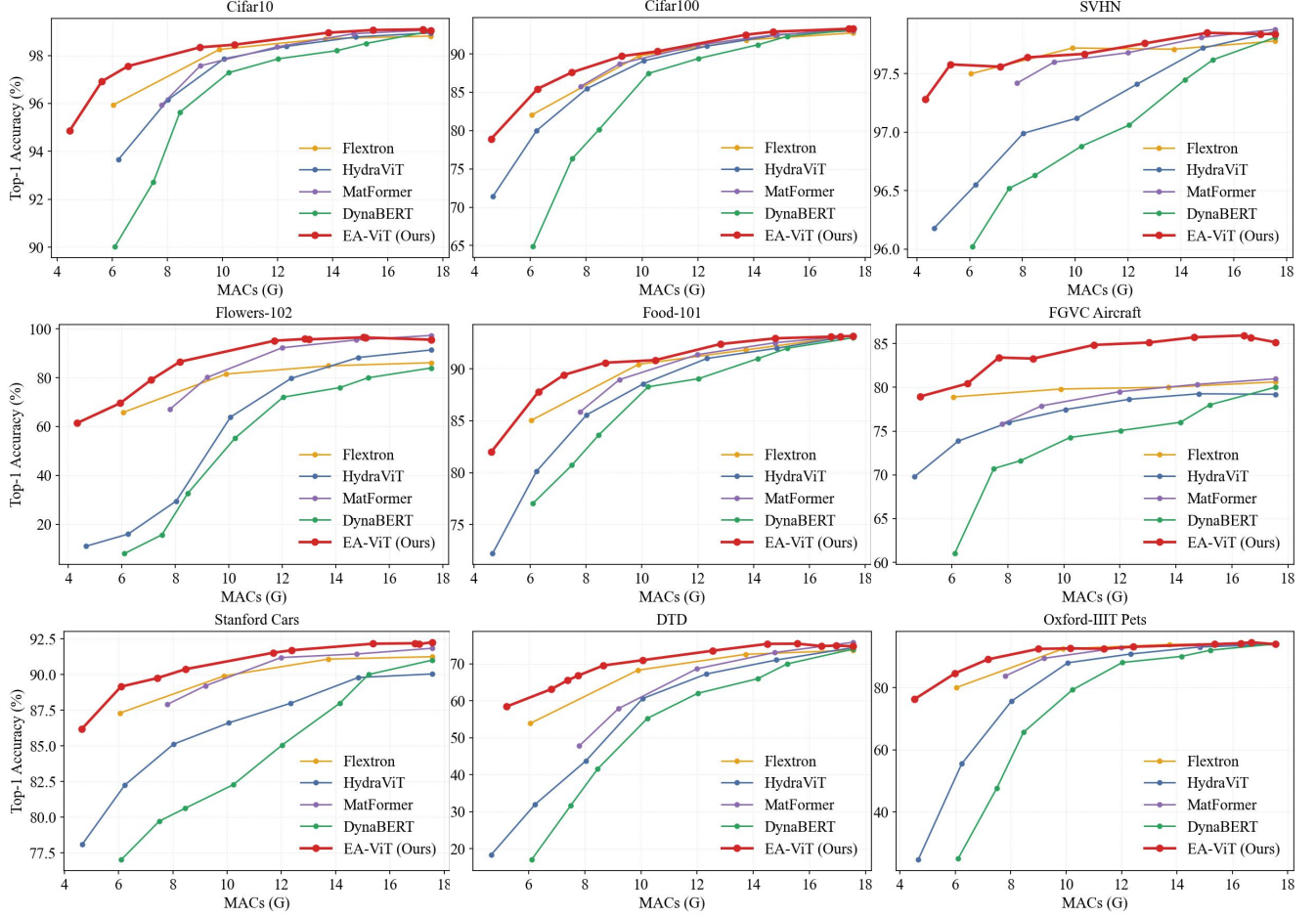
Figure 3. **MACs(G) vs. Top-1 accuracy curves of our method and prior approaches across nine image classification benchmarks.** Our method consistently outperforms existing methods across nearly all MACs constraints on all datasets.

Table 2. **Top-1 accuracy across nine classification benchmarks.** We adopt 8 GMACs constraint. The best scores are in **bold**.

Method	Cifar10 [26]	Cifar100 [26]	SVHN [11]	Flowers [38]	Food101 [1]	Aircraft [35]	Cars [25]	DTD [6]	Pets [39]
DynaBERT [20]	94.24	78.30	96.58	24.53	82.24	71.19	80.19	36.88	57.05
MatFormer [9]	96.17	86.18	97.45	69.00	86.31	76.13	88.10	49.25	84.53
HydraViT [12]	96.11	85.39	96.98	29.27	85.49	75.95	85.07	43.53	75.32
Flextron [3]	97.11	85.95	97.61	73.80	87.79	79.36	88.62	61.21	86.35
EA-ViT (ours)	**97.98**	**88.20**	**97.63**	**85.39**	**90.05**	**83.37**	**90.09**	**67.56**	**90.57**

Since the router's output θ involves discrete architectural decisions, we adopt the Gumbel-Sigmoid [34] trick to enable gradient-based optimization. Specifically, the router's logits are perturbed with Gumbel noise and passed through a temperature-controlled sigmoid to produce a continuous relaxation:

$$y_{\text{soft}} = \text{Sigmoid}\left(\frac{\text{logits} + G_1 - G_2}{\tau}\right), y_{\text{hard}} = \mathbb{I}(y_{\text{soft}} > \delta),$$

(4)

where $G_1, G_2 \sim \text{Gumbel}(0, 1)$ are independent noise sam-

ples, τ is a temperature parameter controlling the approximation sharpness and y_{hard} is obtained by thresholding y_{soft}.

To preserve gradient flow during training, we use the straight-through estimator:

$$y = y_{\text{hard}} - \text{detach}(y_{\text{soft}}) + y_{\text{soft}},$$

(5)

where detach(.) denotes removing gradients during back-propagation. This formulation allows the router to make discrete submodel selections while remaining fully differentiable for end-to-end optimization.
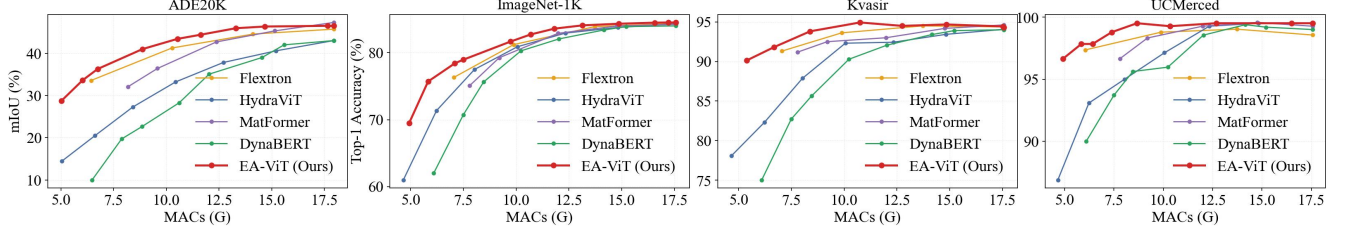
Figure 4. **Comparison of MACs vs. mIoU/Top-1 accuracy between our method and prior approaches on four more challenging datasets:** Semantic segmentation benchmark (ADE20K [61]), a large-scale image classification dataset (ImageNet-1K [8]), a medical image classification dataset (Kvasir [40]), and a remote sensing classification dataset (UCMerced [53]). Our method consistently outperforms existing approaches across nearly all MACs constraints on these more difficult tasks.

Training. To jointly optimize submodel performance and ensure adherence to computational constraints, we formulate a bi-objective minimization problem that balances predictive accuracy with resource usage as follows:

$$\min_{\theta} \left(\mathcal{L}_{\text{CE}}(\theta), \ \left| \frac{\text{MACs}(\theta)}{M_0} - M_t \right| \right), \quad (6)$$

where $M_t \in [0, 1]$ denotes the normalized target MACs, and M_0 is the MACs of the largest model in the design space.

Given a sequence length N and the dimension of each attention head d_{head}, the MACs for a given submodel θ can be computed as:

$$
\begin{aligned}
\text{MACs}(\theta) = \sum_{l=1}^{L} \Big[& D_{\text{MLP}}^{(l)} \cdot 2NE^2 R^{(l)} \\
& + D_{\text{MHA}}^{(l)} \cdot N d_{\text{head}} H^{(l)} \big(4E + 2N \big) \Big].
\end{aligned}
\quad (7)
$$

The first term captures the MLP complexity, while the second term accounts for the computational cost of MHA, including both projection and attention operations.

Let $\theta^*(M_t)$ denote the submodel on the Pareto front discovered by Pareto-Optimal Submodel Search whose MACs most closely match the target constraint M_t. The unified loss function used in Stage 2 is formulated as:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_1 \left(\frac{\text{MACs}(\theta)}{M_0} - M_t \right)^2 + \lambda_2 \left\| \theta - \theta^*(M_t) \right\|_2^2, \quad (8)$$

where λ_1 penalizes deviation from the computational constraint, and λ_2 controls the strength of early-stage supervision based on the reference configuration $\theta^*(M_t)$.

In the early phase of Stage 2, we set a relatively large λ_2 to guide the router toward known Pareto-optimal submodels, thereby accelerating convergence. As training progresses, λ_2 is gradually annealed to zero, allowing the router to autonomously explore the search space and potentially discover submodel configurations that surpass those in the initial Pareto front.

4. Evaluation

4.1. Experiment Setup

Compared Methods. We compare our method against several state-of-the-art approaches that incorporate elastic structures into Transformer architectures. Specifically, we evaluate against DynaBERT [20], which introduces elastic width and depth via knowledge distillation; MatFormer [9], one of the earliest works to explore elasticity in Transformers; HydraViT [12], which builds elasticity based on varying the number of attention heads; and the recently proposed Flextron [3], a powerful framework that integrates elasticity into both the MLP and MHA modules and employs a router for submodel selection. We integrate these methods into the adaptation process to incorporate elastic structures.

Datasets. To comprehensively evaluate the effectiveness and generalizability of our method, we conduct experiments across a wide range of downstream tasks. Specifically, we perform adaptation on nine image classification datasets: Cifar10 [26], Cifar100 [26], SVHN [11], Flowers-102 [38], Food-101 [1], FGVC Aircraft [35], Stanford Cars [25], DTD [6], and Oxford-IIIT Pets [39]. To assess the applicability of our approach to dense prediction tasks, we further evaluate it on one widely-used semantic segmentation benchmarks: ADE20K [61]. We also evaluate our method on the difficult ImageNet-1K [8] datasets. In addition, to demonstrate the robustness of our method in real-world scenarios, we conduct experiments on two real-world application datasets: Kvasir [40] for medical image analysis, and UCMerced [53] for remote sensing applications.

Implementation Details. All experiments are conducted on NVIDIA A100 GPUs. For adaptation and expansion, we adopt ViT-Base as the backbone, initialized with weights pre-trained on ImageNet-21K [44]. During the Curriculum Elastic Adaptation stage, we first trained the model to incorporate elasticity. We then trained the router independently for 1,000 steps, guiding it to output submodels that match

Table 3. **The impact of progressively enabling elasticity across four architectural dimensions:** MLP, MHA, Embed, and Depth. We report the minimum achievable MACs, the highest Top-1 accuracy, and the area under the MACs–accuracy curve (AUC) along with its slope. As more dimensions are made elastic, the architecture achieves better accuracy and broader MACs coverage. Experiments are conducted on Cifar100 [26].

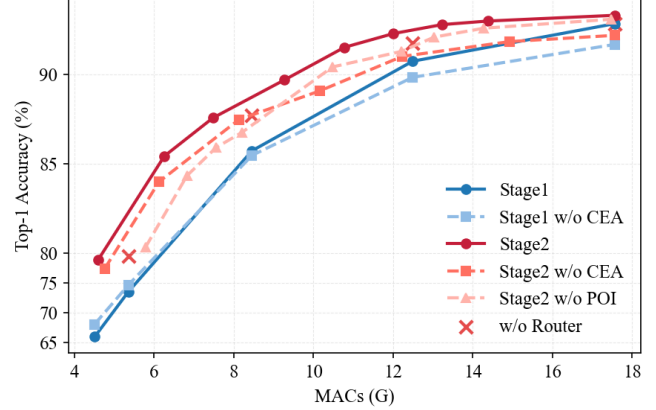| MLP	MHA	Embed	Depth	$MACs_{min}$ ($\downarrow$)	Acc_{max} ($\uparrow$)	Slope ($\downarrow$)	AUC ($\uparrow$)
$\checkmark$	–	–	–	7.80	93.01	1.658	0.505
$\checkmark$	$\checkmark$	–	–	6.04	93.12	1.515	0.646
$\checkmark$	$\checkmark$	$\checkmark$	–	5.11	93.31	1.372	0.651
$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	**4.60**	**93.32**	**1.235**	**0.663**

Figure 5. **The contributions of key components in our framework:** The router, Curriculum Elastic Adaptation (CEA) and Pareto-Optimal Initialization (POI). Experiments are conducted on Cifar100 [26].

the Pareto-optimal configurations identified in the previous search stage. Finally, the router and ViT backbone were jointly optimized. All training phases employed a cosine decay learning rate schedule with linear warm-up. For a fair comparison, all baseline models and our ViT backbone were trained for the same number of epochs on each downstream dataset. Additional training details, including the specific learning rates and number of epochs for each dataset, are provided in the supplementary materials B.

4.2. Comparison with State-of-the-Art Methods

As shown in Figure 3, we conduct a comprehensive comparison between our method and prior state-of-the-art approaches across nine image classification benchmarks. The accuracy–MACs curves clearly demonstrate that our method consistently outperforms existing methods across all datasets and under nearly all computational budgets, with particularly significant advantages in the low-MACs regime. Moreover, we observe that different datasets exhibit varying sensitivities to model size. For simpler datasets such as SVHN [11], accuracy degrades slowly as MACs decrease, indicating a lower dependence on model capacity. In contrast, more complex datasets such as DTD [6] and Flowers-102 [38] experience a steep accuracy drop under constrained MACs, highlighting their greater need for model expressiveness. These observations suggest that our method can adaptively select appropriate submodel sizes based on dataset characteristics, offering a practical advantage in real-world deployment scenarios.

As shown in Table 2, we report the Top-1 accuracy of various methods under an 8 GMACs computational budget. Our method consistently achieves substantial improvements over existing baselines. For instance, on the Flowers-102 [38] dataset, our approach outperforms the best prior method by a remarkable margin of 26.39%, demonstrating its strong ability to handle complex visual patterns under limited computational budgets. Comparison under other MACs constraint can be seen in the supplementary material C.

We further evaluate our method on more challenging tasks, including image segmentation, large-scale image clas-

sification, medical image classification, and remote sensing image classification. As illustrated in Figure 4, our method consistently outperforms previous approaches across all MACs configurations, demonstrating strong generalization capabilities in complex scenarios. These results highlight the versatility and practical value of our approach across diverse real-world applications.

4.3. Ablation Study

To assess the effectiveness of each component in our proposed method, we conduct comprehensive ablation studies.

Multi-Dimensional Elastic Architecture. We investigate whether increasing the number of controllable architectural dimensions leads to improved model performance under varying computational budgets. To this end, we compute four key metrics from the accuracy–MACs curves corresponding to different combinations of elastic dimensions: the minimum achievable MACs, the highest Top-1 accuracy, the average slope, and the Area Under the Curve (AUC). As summarized in Table 3, utilizing only the MLP expansion ratio leads to limited MACs coverage and notable performance degradation, due to the coarse granularity of control along a single architectural dimension. This is reflected in the relatively high slope observed. In contrast, progressively incorporating additional dimensions—such as the number of attention heads, embedding size, and layer depth—consistently increases MACs coverage, improves the maximum achievable accuracy, enhances the AUC, and reduces the slope. These results underscore the importance of multi-dimensional elasticity for robust and efficient submodel deployment across a wide range of computational constraints.

Effectiveness of the Router. As shown in Figure 5, we conduct an ablation study to assess the impact of the router.

We compare submodels manually selected with fixed architectures against those automatically selected by the router, ensuring all models are trained for the same number of epochs. The manually selected submodels consistently lie below the MACs–accuracy curve achieved by the router, indicating inferior performance under comparable computational budgets. These results validate the effectiveness of our routing mechanism in identifying high-quality, resource-efficient submodels.

Curriculum Elastic Adaptation. We conduct an ablation study to evaluate the effectiveness of Curriculum Elastic Adaptation (CEA) during Stage 1 training, as shown in Figure 5. Removing CEA results in the simultaneous training of all submodels from the beginning, which leads to a disproportionately high sampling frequency of smaller submodels. While this slightly improves performance under low-MACs conditions, it negatively impacts the optimization of larger submodels, resulting in noticeable performance degradation at higher computational budgets—likely due to interference in shared parameter learning.

After Stage 2 refinement, models trained with CEA consistently outperform those without it across the entire MACs range. These results underscore the importance of curriculum-based scheduling in promoting training stability and preserving the performance of high-capacity submodels.

Pareto-Optimal Initialization. We conduct an ablation study to assess the impact of Pareto-Optimal Initialization (POI) on the router's performance during Stage 2, as shown in Figure 5. Instead of initializing the router with randomly selected submodels, POI leverages architectures sampled from the Pareto front. This informed initialization allows the router to focus its search on a promising region of the architectural space from the outset. By prioritizing submodels with high potential, POI helps the training process make more effective use of limited computational resources. As a result, the router can more efficiently identify submodels that are both performant and resource-efficient, ultimately leading to improved overall system performance.

4.4. Analysis

We employ t-SNE to visualize the distribution of submodel architectures obtained across nine image classification datasets. As shown in Figure 6, we focus on submodels with MACs in the range of 0.4 to 0.6, and observe that different downstream tasks exhibit distinct structural preferences. Furthermore, the degree of divergence in submodel choices varies significantly across datasets.

Specifically, datasets such as DTD [6] (oriented toward texture recognition) and SVHN [11] (digit classification) demonstrate submodel preferences that deviate markedly from those of general object recognition datasets. This is
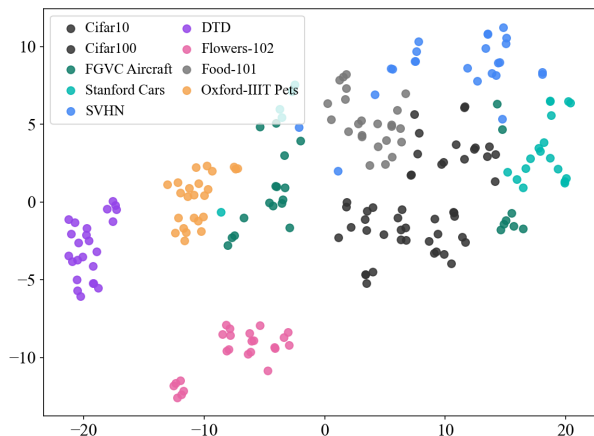

Figure 6. **t-SNE visualization of submodel structures selected by the router.** Experiments are conducted within a normalized MACs range of 0.4 to 0.6 across nine datasets. Submodels from similar datasets tend to cluster closely, while those from dissimilar datasets exhibit distinct structural differences.

reflected by their embeddings forming distant outliers in the t-SNE space. In contrast, structurally similar datasets—such as Cifar10 [26] and Cifar100 [26], or FGVC-Aircraft [35] and Stanford Cars [25]—produce submodels with highly similar configurations, as indicated by the tight clustering of their points. These observations underscore the importance of dataset-specific adaptability in the design of elastic architectures. They further demonstrate the effectiveness of our router in identifying optimal submodel configurations under varying computational constraints across diverse datasets. Detailed architectural specifications for each dataset are provided in the supplementary material D.

5. Conclusion

In this paper, we introduce the first elastic adaptation framework for Vision Transformers, enabling architectural elasticity during the adaptation stage and substantially reducing both training and deployment costs. Our approach features a multi-dimensional elastic ViT design, coupled with a lightweight router that dynamically selects optimal submodels based on different constraints. We incorporate a curriculum learning strategy that gradually expands architectural elasticity during adaptation to ensure stable optimization. Furthermore, we accelerate router convergence by initializing it with Pareto-optimal submodels identified via a customized NSGA-II algorithm, followed by joint training with the backbone. Extensive experiments across a wide range of benchmarks demonstrate that our method consistently outperforms existing approaches and offers new insights into the structural preferences of ViTs under varying resource budgets and downstream task demands. We hope this work will inspire future research on efficient, flexible, and scalable ViT adaptation for real-world deployment.

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014. 5, 6, 15

[2] Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020. 2

[3] Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. In *ICML*, 2024. 2, 5, 6, 15

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1

[5] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *TPAMI*, 2024. 1

[6] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 5, 6, 7, 8, 15

[7] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *TEVC*, 2002. 4

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6, 13

[9] Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. Matformer: Nested transformer for elastic inference. In *NeurIPS*, 2024. 2, 5, 6, 15

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 2, 3

[11] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv:1312.6082*, 2014. 5, 6, 7, 8, 15

[12] Janek Haberer, Ali Hojjat, and Olaf Landsiedel. Hydravit: Stacking heads for a scalable vit. In *NeurIPS*, 2024. 2, 5, 6, 15

[13] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *TPAMI*, 2022. 1

[14] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016. 1

[15] Yizeng Han, Yifan Pu, Zihang Lai, Chaofei Wang, Shiji Song, Junfeng Cao, Wenhui Huang, Chao Deng, and Gao Huang. Learning to weight samples for dynamic early-exiting networks. In *ECCV*, 2022. 3

[16] Yizeng Han, Zeyu Liu, Zhihang Yuan, Yifan Pu, Chaofei Wang, Shiji Song, and Gao Huang. Latency-aware unified dynamic networks for efficient image recognition. *TPAMI*, 2024. 3

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[18] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 1

[19] Charles Herrmann, Richard Strong Bowen, and Ramin Zabih. Channel selection using gumbel softmax. In *ECCV*, 2020. 3

[20] Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dynabert: Dynamic bert with adaptive width and depth. In *NeurIPS*, 2020. 2, 5, 6, 15

[21] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 2

[22] Gagan Jain, Nidhi Hegde, Aditya Kusupati, Arsha Nagrani, Shyamal Buch, Prateek Jain, Anurag Arnab, and Sujoy Paul. Mixture of nested experts: Adaptive processing of visual tokens. In *NeurIPS*, 2024. 3

[23] Samir Khaki and Konstantinos N Plataniotis. The need for speed: Pruning transformers with one recipe. In *ICLR*, 2024. 1

[24] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *CSUR*, 2022. 1

[25] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCVW*, 2013. 5, 6, 8, 13, 15

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5, 6, 7, 8, 15

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *CSUR*, 2017. 2

[28] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *CVPR*, 2021. 3

[29] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *ECCV*, 2022. 1

[30] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 2021. 1

[31] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 2

[32] Yue Ma, Yali Wang, Yue Wu, Ziyu Lyu, Siran Chen, Xiu Li, and Yu Qiao. Visual knowledge graph for human action reasoning in videos. In *ACM MM*, 2022. 14

[33] Yue Ma, Kunyu Feng, Xinhua Zhang, Hongyu Liu, David Junhao Zhang, Jinbo Xing, Yinhan Zhang, Ayden Yang, Zeyu Wang, and Qifeng Chen. Follow-your-creation: Empowering 4d creation through video inpainting. *arXiv:2506.04590*, 2025. 14

[34] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv:1611.00712*, 2016. 5

[35] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv:1306.5151*, 2013. 5, 6, 8, 13, 15

[36] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *AIR*, 2014. 3

[37] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In *CVPR*, 2022. 3

[38] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008. 5, 6, 7, 15

[39] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012. 5, 6, 15

[40] Konstantin Pogorelov, Kristin Ranheim Randel, Carsten Griwodz, Sigrun Losada Eskeland, Thomas de Lange, Dag Johansen, Concetto Spampinato, Duc-Tien Dang-Nguyen, Mathias Lux, Peter Thelin Schmidt, Michael Riegler, and Pål Halvorsen. Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection. In *ACM MMSys*, 2017. 6, 13

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

[42] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *CSUR*, 2021. 2

[43] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *CSUR*, 2021. 2

[44] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *NeurIPS*, 2021. 6

[45] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021. 1

[46] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, 2016. 3

[47] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *ECCV*, 2022. 1

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[49] Zexuan Yan, Yue Ma, Chang Zou, Wenteng Chen, Qifeng Chen, and Linfeng Zhang. Eedit: Rethinking the spatial and temporal redundancy for efficient image editing. *arXiv:2503.10270*, 2025. 14

[50] Suorong Yang, Suhan Guo, Jian Zhao, and Furao Shen. Investigating the effectiveness of data augmentation from similarity and diversity: An empirical study. *Pattern Recognition*, 2024. 14

[51] Suorong Yang, Peijia Li, Xin Xiong, Furao Shen, and Jian Zhao. Adaaugment: A tuning-free and adaptive approach to enhance data augmentation. *TIP*, 2024.

[52] Suorong Yang, Peng Ye, Wanli Ouyang, Dongzhan Zhou, and Furao Shen. A clip-powered framework for robust and generalizable data selection. In *ICLR*, 2024. 14

[53] Yi Yang and Shawn Newsam. Bag-of-visual-words and spatial extensions for land-use classification. In *ACM SIGSPATIAL GIS*, 2010. 6, 13

[54] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *ICLR*, 2019. 2

[55] Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty years of mixture of experts. *TNNLS*, 2012. 3

[56] Wangbo Zhao, Jiasheng Tang, Yizeng Han, Yibing Song, Kai Wang, Gao Huang, Fan Wang, and Yang You. Dynamic tuning towards parameter and inference efficiency for vit adaptation. In *NeurIPS*, 2024. 3

[57] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Zhikai Li, Yibing Song, Kai Wang, Zhangyang Wang, and Yang You. A stitch in time saves nine: Small vlm is a precise guidance for accelerating large vlms. In *CVPR*, 2025. 3

[58] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Hao Luo, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dydit++: Dynamic diffusion transformers for efficient visual generation. *arXiv:2504.06803*, 2025. 3

[59] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dynamic diffusion transformer. In *ICLR*, 2025. 3

[60] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. 1

[61] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 6, 13

[62] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv:2103.11886*, 2021. 1

[63] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. In *NeurIPS*, 2022. 3