International Journal of Computer Vision
https://doi.org/10.1007/s11263-025-02480-w

AutoViT: Achieving Real-Time Vision Transformers on Mobile via Latency-aware Coarse-to-Fine Search

Zhenglun Kong[1] · Dongkuan Xu[2] · Zhengang Li[1] · Peiyan Dong[1] · Hao Tang[1] · Yanzhi Wang[1] · Subhabrata Mukherjee[3]

Received: 2 September 2024 / Accepted: 10 May 2025

Abstract

Despite their impressive performance on various tasks, vision transformers (ViTs) are heavy for mobile vision applications. Recent works have proposed combining the strengths of ViTs and convolutional neural networks (CNNs) to build lightweight networks. Still, these approaches rely on hand-designed architectures with a pre-determined number of parameters. In this work, we address the challenge of finding optimal light-weight ViTs given constraints on model size and computational cost using neural architecture search. We use a search algorithm that considers both model parameters and on-device deployment latency. This method analyzes network properties, hardware memory access pattern, and degree of parallelism to directly and accurately estimate the network latency. To prevent the need for extensive testing during the search process, we use a lookup table based on a detailed breakdown of the speed of each component and operation, which can be reused to evaluate the whole latency of each search structure. Our approach leads to improved efficiency compared to testing the speed of the whole model during the search process. Extensive experiments demonstrate that, under similar parameters and FLOPs, our searched lightweight ViTs achieve higher accuracy and lower latency than state-of-the-art models. For instance, on ImageNet-1K, AutoViT_XXS (71.3% Top-1 accuracy, 10.2ms latency) outperforms MobileViTv3_XXS (71.0% Top-1 accuracy, 12.5ms latency) with 0.3% higher accuracy and 2.3ms lower latency.

Keywords Vision Transformer · Neural Architecture Search · Efficient Inference · Mobile

1 Introduction

Vision Transformer (ViT) (Dosovitskiy et al., 2021) leverages self-attention from the transformer architecture and has achieved state-of-the-art performance in vision tasks such as image classification(Dosovitskiy et al., 2021; Touvron et al., 2021; Chen et al., 2021b), object detection (Dai et al., 2021a; Misra et al., 2021), semantic segmentation (Zheng et al., 2021; Cheng et al., 2021g), image enhancement (Chen et al., 2021c; Lu et al., 2021). While these efforts focus on improving ViT performance, they often compromise model efficiency and portability. ViTs remain relatively unexplored for mobile vision tasks (Mehta et al., 2021) and are difficult to deploy to edge devices due to resource constraints. The popular ViT models are made small by simply scaling them down (Touvron et al., 2021) with a significant reduction in model performance. This challenge can be addressed by the careful manual design of the architecture to meet various resource constraints of different devices. However, this process requires several trials and errors to obtain viable candidates, especially for designing hybrid CNN-Transformer. d'Ascoli et al. (2021); Chu et al. (2021); Dai et al. (2021b); Liu et al. (2021).

While previous works have showcased improvements in on-device efficiency, they often do not prioritize consideration of the real hardware latency when designing the DNN models. Some contain mobile-unfriendly operations that limit performance improvement. LeViT (Graham et al., 2021) employs a convolutional stem in place of the original patch stem. While LeViT successfully reduces FLOPs, it does so at the expense of introducing redundant parameters.

Communicated by Jiaolong Yang.

✉ Zhenglun Kong
kong.zhe@northeastern.edu

[1] Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, USA

[2] North Carolina State University, Raleigh, NC 27695, USA

[3] Microsoft Corporation, Redmond, WA 98052, USA

Published online: 26 May 2025

This becomes a significant constraint for edge devices with limited memory capacity. Besides, relying on FLOPs as a performance metric is misleading since it doesn't necessarily correlate with latency. Additionally, the reshaping process–a data layout change, can cost GPU/NPU cycles, which computation amounts can not be calculated. Also, HardSwish is not inherently supported by the iPhone's CoreML. Sophisticated attention mechanisms, like the ones employed in the Long-Short Transformer , are notoriously hard to support or accelerate on mobile devices.

Neural architecture search (NAS) has shown its benefit over manual design as a powerful technique for the automatic design of neural networks, especially for designing efficient models (Dong et al. 2023a, b; Yu et al. 2025; Li et al. 2021). Autoformer (Chen et al., 2021d) is a dedicated one-shot architecture search framework for pure transformer structures. However, their limited search space leads to performance degradation for very lightweight models. NASViT (Anonymous, 2022) searches for a conv and transformer hybrid structure. However, it has a prolonged training time. The gradient optimization method introduced to circumvent the conflicts between larger and smaller subnets is time-consuming. It also includes mobile unfriendly operations such as the talking head module and window attention. S3 (Chen et al., 2021e) proposes an automatic search space design method to improve the effectiveness of the design space. However, its evolutionary search is time-consuming. Our work aims to improve existing solutions by introducing an efficient, hardware-aware approach. It is optimized to adapt to the target hardware's constraints while meeting specific speed requirements.

To address the existing ViT search problem, we propose an efficient NAS approach that considers three perspectives: *optimal search space, training efficiency, and latency-guided search*. Specifically, we design a new search space that enables the search for hybrid structures of convolution and transformer, which outperform existing hybrid structures, as demonstrated in Fig. 1. To improve training efficiency, we encode the search space into independent supernets, which reduces optimization interference between subnets of different sizes. We propose a weight inheritance strategy, allowing different transformer blocks to share weights for their common parts in each layer during supernet training. Our experiments show that this approach leads to improved performance compared to existing methods.

For edge computing, the selection of models pivots critically on latency. Despite this, existing search algorithms face challenges in accurately evaluating model latency using only FLOPs and parameter counts. As such, a model that appears smaller in size from the search process does not necessarily guarantee higher speed. To overcome the limitations, researchers have attempted to incorporate latency into the search criteria via a latency predictor (Wang et al., 2020). However, developing an accurate predictor is not without challenges-it necessitates hundreds, even thousands, of actual speed tests across diverse model structures to form a robust training dataset. If the search space is expansive, a proportional increase in actual test results is required to ensure accurate predictions across all possible architectures. To this end, we propose a coarse-to-fine search algorithm that utilizes latency as a primary indicator and a parameter as a memory threshold to explore the search space and find the optimal architecture. We leveraged the latency profiling to improve efficiency. As a result of our efforts, we successfully identified models with significantly lower latency than the previous approach of relying on flops and the number of parameters. Our contributions can be summarized as follows:

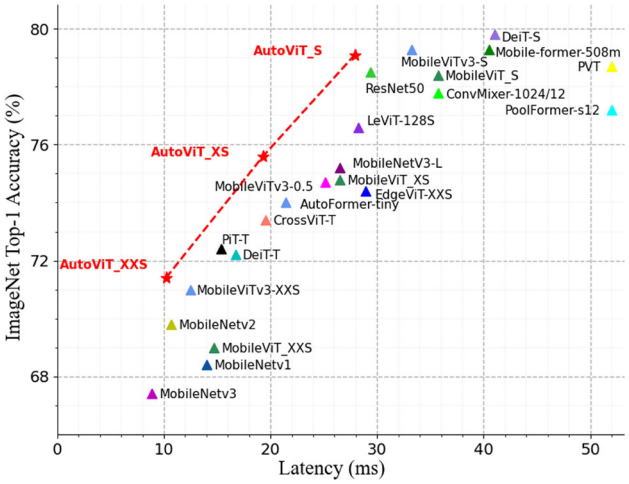

Fig. 1 Top-1 accuracy and sizes of different models on ImageNet. Our method achieves a better trade-off than other efficient CNN and transformer-based models

- **Hybrid Search Space:** We designed a search space that combines CNN and transformers, leveraging the strengths of both, to create lightweight ViTs suitable for mobile vision tasks.
- **Multi-Size Supernet Training Scheme:** To tackle the large search space in vision transformers and optimize subnets of different sizes, we use a multi-size supernet training scheme to reduce optimization interference between subnets and apply weight inheritance for efficient training.
- **Latency-aware Search Scheme:** We revisit the design principles of ViT and its variants through latency analysis and propose a latency profiling model, which can efficiently estimate the latency of network candidates. We propose a parameter-latency-oriented coarse-to-fine search strategy to find the optimal subnets for hardware deployment.
- **Performence:** Our searched model can achieve up to 79.2% accuracy on ImageNet-1k with 6.3M parameters

and 1.4 GFLOPs, exceeding the performance of existing models with similar resource budgets.

2 Related Work

2.1 Vision Transformers

ViT (Dosovitskiy et al., 2021) is a pioneering work that uses only transformer blocks to solve various vision tasks. Compared to traditional CNN structures, ViT allows all the positions in an image to interact through transformer blocks. In contrast, CNNs operate on a fixed-sized window with restricted spatial interactions, which hinders their ability to capture relations at the pixel level in both spatial and time domains. Since then, many new variants have been proposed. For example, DeiT (Touvron et al., 2021), T2T-ViT (Yuan et al., 2021b) tackle the data-inefficiency problem in ViT by training only with ImageNet. PiT (Heo et al., 2021) replaces the uniform structure of the transformer with a depth-wise convolution pooling layer to reduce spacial dimension and increase channel dimension. SViTE (Chen et al., 2021f) alleviates training memory bottleneck and improves inference efficiency by co-exploring token and attention head sparsity. There is also a series of papers that improve the efficiency of vision transformer training (Xin et al. 2024a, b, c).

2.2 Neural Architecture Search

Neural Architecture Search automates the design of deep learning models. One-shot NAS methods train a single supernet, from which multiple architectures can be sampled and evaluated without training each one from scratch (Poyser & Breckon, 2024). This eliminates the need for independent training of every candidate model, making NAS more computationally feasible. A key advancement in one-shot NAS is weight sharing (Pham et al., 2018). Instead of training individual architectures, a supernet is trained to convergence, and subnets inherit pre-trained weights, allowing for rapid evaluation. Once-for-All (Cai et al., 2019) introduced a progressive shrinking strategy, which first trains the largest subnetworks and gradually fine-tunes smaller ones. SPOS (Guo et al., 2020) improved this process by incorporating evolutionary search, achieving more optimal selections. Despite the acceleration provided by one-shot NAS, the computational cost of architecture evaluation remains significant, particularly when searching across large design spaces. ProxylessNAS (Cai et al., 2018) avoids full supernet training by subsampling only a single path through the directed acyclic graph (DAG) per iteration, significantly reducing memory and computation overhead. Supernet-based NAS approaches further refine supernet training and selection. BigNAS (Yu et al., 2020) addresses the optimization-gap issue by introducing regularization and novel initialization strategies to improve weight sharing. AlphaNet (Wang et al., 2021a) enhances generalization through alpha-divergence regularization. Other methods explore search space pruning (Bender et al., 2020; Xia et al., 2022; Wan et al., 2022) to reduce redundant candidate architectures without sacrificing accuracy. AttentiveNAS (Wang et al., 2021b) builds upon BigNAS by introducing a Pareto-aware architecture search strategy, reducing search space size while maintaining performance.

2.3 Efficient ViT Design

There has been an increasing emphasis on developing efficient ViT through NAS-driven designs and hand-crafted approaches. Each method has showcased efficiency gains, as evidenced by hardware evaluations. Regarding hand-crafted designs, LeViT (Graham et al., 2021) replaces the original patch stem with a convolutional stem, enhancing inference speed for image classification (Xin et al. 2023, 2024d). Regarding NAS-driven designs, HAT (Wang et al., 2020) searches for an encoder-decoder transformer structure and requires retraining or finetuning of the optimal candidates obtained during the search. Autoformer (Chen et al., 2021d) entangles the weights of different vision transformer blocks within the same layer during supernet training and trains three supernets at different scales to reduce training time. NASViT (Anonymous, 2022) expands the search space to include both convolutions and transformers. However, these approaches have limitations, including the use of operations that are not mobile-friendly, over-reliance on parameters and FLOPs as evaluation metrics, restrictive search spaces, and prolonged training or search durations.

Our advantages over existing works can be summarized as follows: ① Unlike hand-crafted methods, leveraging network search allows pinpointing the optimal structure tailored to specific hardware constraints. Hand-crafted designs yield models that primarily differ in dimensions. While they may serve as general-purpose models suitable for a broad range of devices, they may not be as finely optimized for a specific hardware profile as network-searched models. We incorporated a hybrid search space with an inductive bias. This not only broadens the search space but also accelerates convergence and mitigating local optima. ② From a hardware-oriented perspective, it is more efficient to define a narrow search space tailored to the specific constraints of target devices. By segmenting the solution into multiple supernets, we cannot only sidestep conflicts but also significantly reduce training overhead. ③ Instead of the traditional approach of relying on hardware deployment for every candidate or using a latency predictor, our method stands out in its accuracy and efficiency. Our latency prediction model is a training-free theoretical model suitable for general-purpose hardware, GPU. It considers the properties of the target hard-

ware, the model type, the model size, and the data granularity. It then quantitatively captures both the computation latency and data movement latency.

3 Methodology

3.1 Background

We aim to search for lightweight and high-performance models for deployment on edge devices. There are two problems with current ViT models: (i) Most of them sacrifice efficiency to improve accuracy. The large computation cost and the number of model parameters make it difficult to deploy these models to devices such as cell phones or FPGAs (Field Programmable Gate Arrays). (ii) Most of the them are done by simply scaling down from the original dimension to obtain models of different sizes (Dosovitskiy et al., 2021; Touvron et al., 2021). This coarse-grained model selection significantly sacrifices the accuracy of small models and offers limited flexibility in adjusting the size. Despite superior performance in the high computational budget regime, ViTs still do not perform as well as their CNN counterparts on small or medium-sized architectures, especially compared to CNN architectures highly optimized by neural architecture search. CNN networks such as MobileNets (Howard et al., 2017), ShuffleNetv2 (Ma et al., 2018), ESPNetv2 (Mehta et al., 2019), and MNASNet (Tan et al., 2019) can easily replace the heavyweight backbones in existing task-specific models to reduce the model size and improve latency. One major drawback is that they are spatially localized. On the other hand, the transformer can learn global representations but ignores the spatial induction bias inherent in CNNs and thus requires more parameters to learn visual representations.

Based on these considerations, we focus on combining the advantages of CNN (e.g., spatial induction bias) and ViT (e.g., input adaptive weighting and global processing) to find a hybrid architecture that considers both performance and efficiency. Past work (Mehta et al., 2021) has shown that incorporating the downsampling mechanism of convolution into the transformer architecture can effectively reduce the size of the model and its ability to process high-resolution images, which can be of great benefit to the learning and deployment of the transformer. Due to the efficiency of local computation, convolution is introduced to process high-resolution inputs, while the transformer is used to process low-resolution features to extract global information.

3.2 Search Space

As shown in Fig. 2, each hybrid block consists of convolution layers and a mobile block. MBConv refers to MobileNetV2 (Sandler et al., 2018) blocks, which reduce

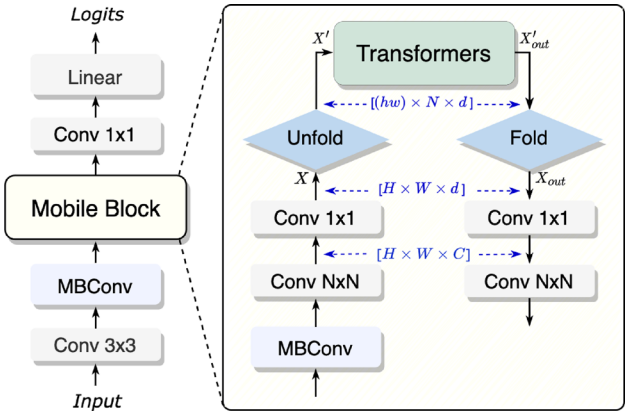

Fig. 2 Our hybrid structure. Conv and MBConv refer to standard convolution and inverted residual blocks, respectively. All CNN and transformer blocks contain a stack of dynamic layers with searchable configurations

computation through depthwise separable convolutions and inverted residuals, enabling low-cost feature extraction before Transformer blocks. It also minimizes on-device execution bottlenecks by avoiding memory-heavy tensor reshaping. Additionally, it efficiently downsamples feature maps, allowing Transformers to focus on low resolution global features, resulting in a better accuracy-latency tradeoff. Standard ViT reshapes input tensor X of size $H \times W \times C$ into $N \times d$. Where C, H, and W represent channels, height, and width, N represents the number of patches and d is the dimension. However, reshaping into a 2D feature ignores the spatial inductive bias that is inherent in CNNs. Following (Mehta et al., 2021), we apply a $n \times n$ standard convolution layer followed by a point-wise convolution layer to produce X. The convolution layer encodes local spatial information while the point-wise convolution projects the tensor to a high-dimensional space by learning linear combinations of the input channels. To learn global representations with spatial inductive bias, we reshape (unfold) X into N non-overlapping flattened patches X' of size $hw \times N \times d$, where hw is the number of pixels of one patch P. The folding-unfolding process replaces local processing in convolutions with global processing using transformers. This allows the transformer block to have CNN- and ViT-like properties, which helps it learn better representations with fewer parameters and simple training recipes. For each patch, inter-patch relationships are encoded by applying transformers for each pixel to obtain $X'_{out}(p) = Transformer(X'(p))$, where $1 \leq p \leq P$. This transformer block prevents ViT from losing either the patch order or the spatial order of pixels within each patch. After that, we reshape (fold) X'_{out} back to X_{out} of size $H \times W \times C$. The spatial order of pixels within each patch is retained throughout the process.

We summarize the detailed dimensions of our search space. In Tab. 1, width represents the channel size for CNN

Table 1 Our search space: The search space is divided into three individual supernets within different parameter ranges to satisfy different resource constraints

Search Dimension	Width	Depth	Number of Heads	Expansion ratio
Supernet XXS				
Conv	{16, 24}	-	-	-
MBConv-1	{16, 24}	{1, 2}	-	1
MBConv-2	{24, 32}	{2, 3}	-	1
MBConv-3	{32, 48}	{2, 3}	-	1
Transformer-1	{48, 64}	{2, 3, 4, 5}	{2, 3, 4}	{1.5, 2}
Transformer-2	{64, 80}	{2, 3, 4, 5}	{2, 3, 4}	{1.5, 2}
Transformer-3	{80, 96}	{2, 3, 4, 5}	{2, 3, 4}	{1.5, 2}
Transformer-4	{96, 112}	{2, 3, 4, 5}	{2, 3, 4}	{1.5, 2}
MBPool	{1000, 1280}	-	-	6
Number of Stage		{3, 4}		
Params Range		$1 \sim 1.8$M		
Supernet XS				
Conv	{16, 24}	-	-	-
MBConv-1	{24, 32}	{1, 2}	-	1
MBConv-2	{32, 48}	{2, 3}	-	1
MBConv-3	{48, 64}	{2, 3}	-	1
Transformer-1	{64, 80}	{2, 3, 4, 5}	{4, 5, 6}	{1.5, 2}
Transformer-2	{80, 96}	{2, 3, 4, 5}	{4, 5, 6}	{1.5, 2}
Transformer-3	{96, 112}	{2, 3, 4, 5}	{4, 5, 6}	{1.5, 2}
Transformer-4	{112,128}	{2, 3, 4, 5}	{4, 5, 6}	{1.5, 2}
MBPool	{1000, 1280}	-	-	6
Number of Stage		{3, 4}		
Params Range		$1.6 \sim 2.6$M		
Supernet S				
Conv	{16, 24}	-	-	-
MBConv-1	{24, 32}	{1, 2}	-	1
MBConv-2	{32, 48}	{2, 3}	-	1
MBConv-3	{48, 64}	{2, 3}	-	1
Transformer-1	{64, 96}	{2, 3, 4, 5}	{7, 8, 9}	{1.5, 2}
Transformer-2	{96, 128}	{2, 3, 4, 5}	{7, 8, 9}	{1.5, 2}
Transformer-3	{128, 160}	{2, 3, 4, 5}	{7, 8, 9}	{1.5, 2}
Transformer-4	{160, 192}	{2, 3, 4, 5}	{7, 8, 9}	{1.5, 2}
MBPool	{1280,1560}	-	-	6
Number of Stage		{3, 4}		
Params Range		$4.0 \sim 7.2$M		

layers and hidden dimension for transformer layers, respectively. The depth denotes the number of repeated CNN and transformer layers for each block. The expansion ratio refers to the expansion ratio of the depth-wise convolution layer for CNN layers and the MLP expansion ratio for transformer layers. For each CNN block, we search for the optimal channel widths, block depths, expansion ratios, and kernel sizes. For each transformer block, we search for the best number of windows, hidden feature dimensions, depths, and MLP expansion ratios. Following one-shot NAS methods, we encode the search space into a supernet. All subnets share the weights of their common parts. The supernet is the largest model in the space. In particular, the supernet stacks the maximum number of transformer blocks with the largest embedding dimension, Q-K-V dimension, and MLP ratio as defined in the space. During training, all possible subnets

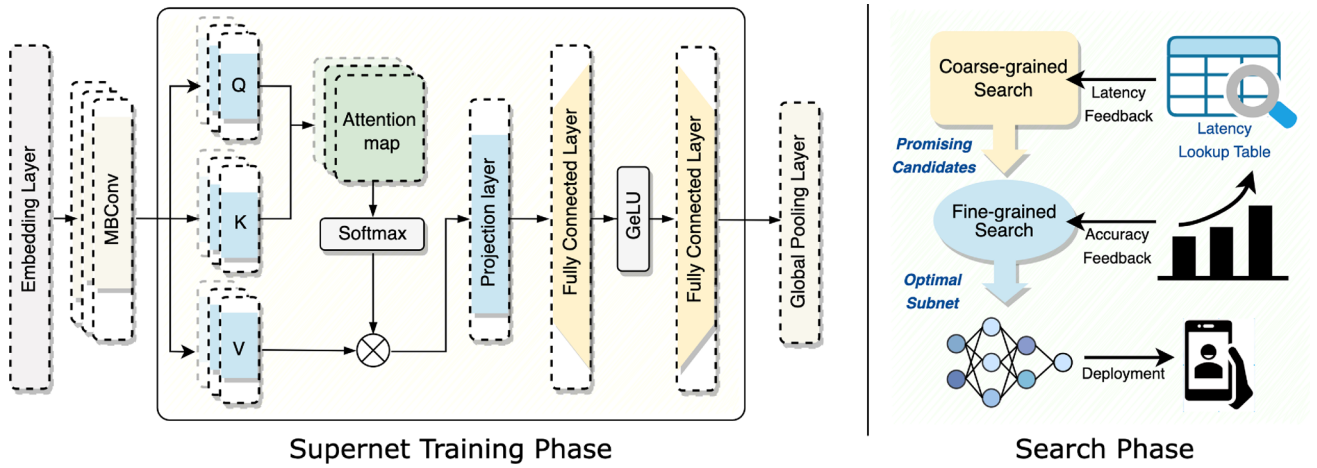

Fig. 3 Framework Overview. We train a weight-shared supernet by iteratively optimizing randomly sampled subnets. *Supernet Training Phase:* We search for the number of heads and expansion ratio of a transformer block, along with the width and depth of MBConv. Each layer and block are dynamic. Solid lines and dark blocks represent selected components in contrast to the dashed lines and lighter blocks. *Search Phase:* Perform a coarse-to-fine search with hardware latency constraints to find the model with the highest validation accuracy

are uniformly sampled, and the corresponding weights are updated.

Our transformer block structure, along with the search space allows us to search for lightweight models. This is mainly due to learning of global representations with transformers. For a given patch, prior work converts spatial information by learning linear combinations of pixels. The global information is then encoded by learning inter-patch information using transformers. As a result, these models lose the image-specific inductive bias, which is inherent to CNNs. As a result, they require more capability to learn visual representations. Hence, they are deep and wide. Unlike these models, our model uses convolution and transformers in such a way that the resulting transformer blocks have convolution-like properties while allowing for global processing. This modeling capability allows us to design shallow and narrow models that are lightweight.

According to the constraints on model parameters, we partition the large-scale search space into three sub-spaces based on parameters and encode them into three independent supernets. Such a partition allows the search algorithm to concentrate on finding models within a specific parameter range, which can be specialized by users according to their available resources and application requirements. We set overlapping parameter sizes between different scales to maintain continuity. Rather than training completely disjoint models, our weight inheritance strategy allows smaller subnets to inherit shared parameters from larger supernets, preserving cross-subnet dependencies. It also reduces gradient conflicts between large and small sub-networks trained via weight-sharing due to gaps in model sizes. Additionally, our search space is flexible, enabling adaptation to different hardware constraints beyond predefined partitions.

3.3 Neural Architecture Search Pipeline

We apply one-shot NAS, which includes two phases: (i) Supernet training: A supernet is trained to encompass all candidate subnetworks, optimizing their parameters simultaneously through weight sharing. (ii) Subnet search: The best sub-network is selected from the trained supernet under various resource constraints (Cai et al., 2019). Typical search techniques include evolutionary algorithms (Guo et al., 2020). Fig. 3 illustrates our framework. We adopt the search space defined in Tab. 1 and train a weight-shared supernet by iteratively optimizing randomly sampled subnets. The search optimizes width, depth, expansion ratio, and the number of self-attention heads for both CNN and transformer layers, with dynamic layer configurations in each block. After training the supernet, we perform a coarse-to-fine search with hardware latency constraints to find the model with the highest validation accuracy. We propose a latency prediction model that accounts for network properties, hardware memory access patterns, and parallelism levels.

3.4 Supernet Training

The classical supernet training strategy encounters the following challenges in transformer search space: (i) Slow convergence. This can be attributed to the independent training of transformer blocks resulting in the weights being updated a limited number of times (Chen et al., 2021d). (ii) Low performance. The performance of the subnets that

inherit weights from the one-shot supernet, trained under classical weight sharing strategy, is far below their performance of training from scratch. This limits the ranking capacity of supernets (Anonymous, 2022). To mitigate this, existing works perform additional retraining of the searched architectures since their weights are not fully optimized.

Weight Inheritance Inspired by Autoformer (Chen et al., 2021d) operating on transformer-only search space, we propose a weight inheritance strategy, allowing different transformer blocks to share weights for their common parts in each layer during supernet training. In the implementation, for each layer, we only need to store the weights of the largest of the n homogeneous candidate blocks. The remaining smaller building blocks can extract the weights directly from the largest building block. Formally, let W_l denote the weights of the largest block in layer l, and let W_l^i denote the weights of the i-th candidate block in the same layer. The weight inheritance is expressed as: $W_l^i = W_l[: d_i^1, : d_i^2, \ldots, : d_i^k]$, where d_i^j represents the dimension of the i-th candidate block along the j-th axis, and k is the number of axes. To quantify the effectiveness of this sharing mechanism, we define a weight sharing efficiency metric:

$$E = \frac{1}{N} \sum_{n=1}^{N} \frac{w_{\text{shared}}}{w_{\text{total},n}}, \tag{1}$$

where N is the total number of candidate blocks across layers, w_{shared} is the number of parameters inherited (shared) from the largest block, and $w_{\text{total},n}$ is the total number of parameters in the n-th candidate block. In each training iteration, we randomly sample a transformer architecture. We then obtain its weights from the supernet and compute the losses of the corresponding subnets. Finally, we update the sampled subnet's weights while freezing the remaining ones. The architecture search space $\mathcal{S}$ is encoded within the supernet.

The second stage is to search for subnet architectures by ranking their performance based on the learned weights. The objective is to select:

$$s^* = \arg\max_{s \in \mathcal{S}} A\big(\mathcal{S}(s, W^*)\big) \quad \text{subject to} \quad L(s) \le L_{\max}, \tag{2}$$

where $A(\cdot)$ indicates the top-1 accuracy on the validation dataset, W^* denotes the optimized weights of the supernet, s is a candidate subnet, $L(s)$ is its latency, and $L_{\max}$ is the maximum allowed latency. Alternatively, to jointly account for latency and computational cost, we formulate the objective as a cost minimization over a reduced (inductively biased) search subspace $\mathcal{S}'$:

$$s^* = \arg\min_{s \in \mathcal{S}'} \left[L(s) + \lambda \cdot C(s) \right], \tag{3}$$

where $C(s)$ represents an auxiliary computational cost metric (e.g., parameter count or FLOPs) and λ controls the trade-off between latency and cost. (Here, $\mathcal{S}' \subset \mathcal{S}$ denotes the candidate architectures that conform to our inductive bias-for example, those with gradually increasing widths across transformer stages.) The supernet weights W are optimized during training by:

$$W^* = \arg\min_{W} \mathbb{E}_{s \sim \mathcal{U}(\mathcal{S})} \left[\mathcal{L}\big(\mathcal{S}(s, W)\big) \right], \tag{4}$$

where $\mathcal{L}(\cdot)$ represents the loss function on the training dataset and $\mathcal{U}(\mathcal{S})$ is a uniform distribution over the search space. This expectation is typically approximated via mini-batch stochastic gradient descent:

$$W_{t+1} = W_t - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla_W \mathcal{L}\big(\mathcal{S}(s_i, W_t)\big), \tag{5}$$

where η is the learning rate, t is the iteration index, B is the batch size, and $s_i \sim \mathcal{U}(\mathcal{S})$ are the sampled subnets. Our weight inheritance strategy promotes stability and convergence in supernet training by allowing smaller candidate subnets to inherit a robust, pre-optimized weight configuration from the largest block. This shared initialization reduces variance across subnet parameters, ensuring that even the smaller subnets benefit from a strong starting point rather than random initialization, which improves gradient flow during backpropagation. Compared with previous weight-sharing methods, our strategy has three advantages: (i) Faster convergence. Weight inheritance allows each block to be updated more times than the previous independent training strategy. (ii) Low memory cost. We store only the largest building blocks' parameters for each layer instead of all the candidates in the space.

Training Efficiency. We improve training efficiency in two ways. ① *Search space reduction*: We introduce an inductive bias where the model dimension (width) increases gradually for each transformer stage, resulting in candidates that do not follow the pattern being discarded. This reduces the search space from 10^{16} to 10^{10} subnets and improves training efficiency. For reference, AutoFormer (Chen et al., 2021d) has 10^{16} and BigNAS (Yu et al., 2020) has more than 10^{12} subnets. Similar architectural inductive bias is applied in the stage-based PVT (Wang et al., 2021c) and the Swin Transformer (Liu et al., 2021). ② Weight inheritance allows different transformer blocks to share weights of their common parts in each layer with significant benefits of faster convergence, and improved subnet performance.

3.5 Latency-aware Search Scheme

Upon obtaining the trained supernet, we run a search algorithm to select the optimal subnets. Existing strategies optimize the inference speed of transformers via computational complexity (MACs) or throughput (images/sec) derived from server GPUs. Such metrics, however, fail to accurately reflect real on-device latency. Traditional hardware-aware network search methods usually depend on the hardware deployment of each candidate within the search space to ascertain latency - a process that is both time-consuming and inefficient. A single candidate demands hundreds of inferences to generate an accurate latency, prolonging the search process. Existing works(Wang et al., 2020) employ a latency predictor, pre-trained with thousands of real latency data points, as an offline method to forecast the candidate latency, as opposed to obtaining real latency by inference during the search. This method, however, is only applicable to a relatively small search space. For larger search spaces, an increased volume of measured latency data is required as a training set for the predictor, substantially raising the time cost. If the test set is inadequate, the predictor fails to estimate the latency accurately.

Latency Profiling To address these challenges, we construct a latency lookup table by collecting the on-device latency of MBConv and transformer blocks of varying dimensions. Specifically, the Conv width includes {16, 24}, the MBConv width includes {16, 24, 32, 48}, the Transformer block width includes {48, 64, 80, 96, 112}, the number of heads includes {2, 3, 4, 5}, and the expansion ratio includes {1.5, 2}-resulting in a total of 46 modules. Note that the speed of an individual module executed in isolation may differ from that in a full network due to inter-module interactions. Therefore, we measure the latency of each module based on the accuracy drop observed when it is removed from the entire model, providing a more realistic assessment of its latency impact. In addition, we model the latency of a module m using a linear combination of its operator and design contributions:

$$L(m) = \sum_{i=1}^{N_o} l_i \cdot o_i(m) + \sum_{j=1}^{N_d} t_j \cdot d_j(m), \tag{6}$$

where $L(m)$ is the latency of module m, l_i and t_j are latency coefficients, $o_i(m)$ denotes the frequency (or cost) of the i-th operator in m, $d_j(m)$ represents the j-th design parameter (e.g., channel width or expansion ratio), and N_o and N_d are the total numbers of operator types and design parameters, respectively.

Let $L(m_i)$ be the latency of module m_i, and let $A(M)$ be the accuracy of the entire model M. We define the latency impact of module m_i as:

$$L_{\text{impact}}(m_i) = \frac{A(M) - A(M \setminus m_i)}{L(M) - L(M \setminus m_i)}, \tag{7}$$

where $M \setminus m_i$ is the model with module m_i removed.

Algorithm 1 Latency-aware Coarse-to-fine Search.

Coarse-grained latency guided search:
Input: Latency space $\mathcal{L}$, Search space $\mathcal{P}$, Supernet $\mathcal{S}$, Latency budget $\mathcal{T}_l$, Parameter budget $\mathcal{T}_p$
for i in N epochs **do**
  Randomly sample one subnet architecture α_i from $\mathcal{S}$
  Obtain size and latency through $\mathcal{L}$
  if Budget of $\alpha_i \leq \mathcal{T}_l \& \mathcal{T}_p$ **then**
    Save subnet to list $\mathcal{A} \leftarrow \{\alpha_i\}$
    $\mathcal{A} :=$ the Top K candidates;
  end if
end for
Output Promising subnet candidates $\mathcal{A}$

Fine-grained accuracy guided search:
Input: Number of generation iteration T, validation dataset D_{val}, mutation probability of depth
Output: The optimal subnet α. P_d, mutation probability of each layer P_m
while search step $t \in (0, T)$ **do**
  while $\alpha_i \in \mathcal{A}$ **do**
    Obtain the accuracy of the subnet $N(\alpha_i, W_{\alpha_i})$ on D_{val}
  end while
  $Top_k :=$ the Top K candidates;
  $Pop_C =$ Crossover(Top_k, S, C)
  $Pop_M =$ Mutation(Top_k, P_d, P_m, S, C)
  $Pop(t+1) = Pop_C \cup Pop_M$
end while

Search Pipeline We propose a coarse-to-fine strategy, which involves integrating latency feedback directly into the design loop when searching for models. This eliminates the need for FLOPs as the latency proxy and reduces the searching time, enabling us to design specialized models efficiently for various hardware platforms.

Fig. 3 on the right illustrates our coarse-to-fine strategy. It contains two steps: Initially, we aim to identify a rough skeleton of promising network candidates. Thereafter, we sample multiple fine-grained variations surrounding each skeleton architecture of interest. More specifically, during the coarse-grained phase, we execute a network search using parameters such as memory threshold and perform latency evaluations using the lookup table. Candidates that meet our latency budget are selected to proceed to the fine-grained search phase. Here, we conduct an evolutionary search to find the optimal subnets (as shown in Fig. 4). Our objective is to maximize accuracy while minimizing the model size:

$$\max_{s \in \mathcal{S}} \{A(s), -S(s)\} \quad \text{subject to} \quad L(s) \leq L_{\max}, \tag{8}$$

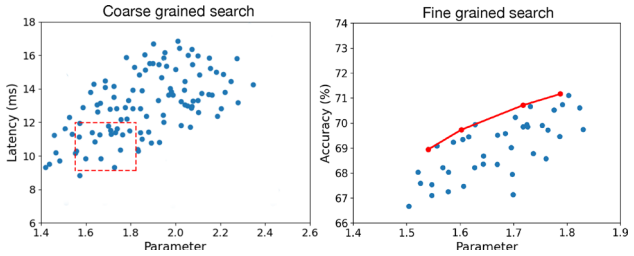

Fig. 4 Coarse-to-fine Architecture Search. *Left:* Pre-define some promising candidate network skeletons by the number of parameters and latency budget. *Right:* Generate more architectures by randomly mutating each skeleton architecture by varying the depth slightly and using the weights from the single-stage model for the induced child models to evaluate all the candidates

where $A(s)$ is the accuracy of subnet s, $S(s)$ is its model size, $L(s)$ is its latency, and $L_{\max}$ is the maximum allowed latency.

At the onset of the evolutionary search, we select N random architectures as seeds. The top-k architectures are chosen as parents to generate the next generation via crossover and mutation. For a crossover, two randomly selected candidates are chosen and crossed to produce a new one during each generation. For mutation, a candidate first modifies its depth with a certain probability P_d, followed by mutating each block with a probability P_m to generate a new architecture.

Let s_t^i be the i-th subnet in generation t. The evolutionary process can be described as:

$$s_{t+1}^j = \begin{cases} C(s_t^k, s_t^l) & \text{with} P_c \\ M(s_t^i) & \text{with} 1 - P_c \end{cases} \tag{9}$$

where P_c is the crossover probability, $C(\cdot, \cdot)$ is the crossover operation, $M(\cdot)$ is the mutation operation, and k and l are randomly selected indices.

Different devices interpret and execute specific operations distinctively due to variations in their underlying architecture, design, computational capabilities, memory management, IO, and bandwidth. Therefore, an operator's latency profile on one hardware platform (e.g., a mobile CPU/GPU) may differ considerably from its profile on another, such as a distinct GPU. This variance stems from the unique characteristics of each hardware. Consequently, it becomes imperative to conduct device-specific optimizations and tests. One cannot simply port results from one hardware setting to another and expect them to remain valid. Nevertheless, crafting multiple lookup tables, although demanding, is substantially more efficient than undertaking multiple comprehensive testing cycles. In our approach, we require data from under 100 instances, while traditional methods might need data from 1000+ instances. Additionally, it is training-free, while conventional techniques require training, increasing resource consumption.

To account for hardware-specific variations, we can define a hardware-specific latency function: $L_h(s) = f_h(s, \boldsymbol{\theta}_h)$, where h denotes a specific hardware platform, s is the subnet, and $\boldsymbol{\theta}_h$ are the parameters of the latency lookup table for hardware h. Alg. 1 shows the procedure of our latency-aware coarse-to-fine search method.

4 Experiments

4.1 Datasets and Implementation Details

Our experiments are conducted on ImageNet-1K (Deng et al., 2009), with approximately 1.2 million images. We report the accuracy of the 50k images in the test set. The image resolution is 256×256.

We train the supernets using a similar recipe as DeiT. Data augmentation techniques, including RandAugment (Cubuk et al., 2020), Cutmix (Yun et al., 2019), Mixup (Zhang et al., 2017) and random erasing (Zhong et al., 2020), are adopted with the same hyperparameters as in DeiT except for the repeated augmentation. Images are split into patches of size 16×16. All the models are implemented using PyTorch 1.7 and trained on V100 GPUs. In the evolutionary search process, we configure a population size of 50 and proceed through 20 generations. Within each generation, we select the top-performing 10 architectures to function as parents, which then produce offspring networks via mutation and crossover mechanisms. We assign mutation probabilities for P_d and P_m as 0.2 and 0.4.

4.2 Experimental Results

We conducted an architecture search on ImageNet and obtained several hybrid transformer models with different parameter sizes. All these models directly inherit the weights from the supernet without additional retraining and other post-processing. We present a summary of performance in Tab. 2. We compare our hybrid models with various models, namely: MobileNets (Howard et al., 2017), ShuffleNetv2 (Ma et al., 2018), ESPNetv2 (Mehta et al., 2019), MobileViT (Mehta et al., 2021), DeiT-T (Touvron et al., 2021), T2T-T (Yuan et al., 2021b), PiT-T (Heo et al., 2021), CrossViT-T (Chen et al., 2021b), DenseNet-169 (Huang et al., 2017), Efficient-Net-B (Tan & Le, 2019), LocalViT-T (Li et al., 2021), CeiT-T (Yuan et al., 2021a), PVT (Wang et al., 2021c), LeViT (Graham et al., 2021), NASViT (Anonymous, 2022), GLiT (Chen et al., 2021a), PoolFormer (Yu et al., 2022), EfficientFormer (Li et al., 2022) and Moat (Yang et al., 2022). As shown in Fig. 1, our AutoViT_XXS, with only 1.8M parameters and 0.3G FLOPs, achieves a Top-1 accuracy of

Table 2 Accuracy comparison on ImageNet-1K with state-of-the-art CNN and transformer-based models under similar parameters and computational cost

Model	Params (M)	FLOPS (*G*)	Top-1 Accuracy (%)	Latency (ms)
MobileNetv3	2.5	0.1	67.4	8.9
ShuffleNetv2	2.3	0.1	69.4	-
MobileNetv2	2.6	0.2	69.8	10.7
ESPNetv2	2.3	0.2	69.2	-
MobileViT_XXS	1.3	0.2	69.0	14.7
MobileNetV3-L	5.4	0.2	75.2	26.5
MobileViTv3-XXS	1.2	0.3	71.0	12.5
MobileNetv1	2.6	0.3	68.4	14.0
LeViT-128S	7.8	0.3	76.6	28.2
EfficientNet-B0	5.3	0.4	76.3	14.5
LeViT-128	9.2	0.4	78.6	36.8
Mobile-former-508m	14.0	0.5	79.3	40.5
EdgeViT-XXS	4.1	0.6	74.4	28.9
MobileViT_XS	2.3	0.6	74.8	26.5
PiT-T	10.6	0.7	72.4	15.4
tiny-MOAT-0	3.4	0.8	75.5	-
T2T-T	4.3	1.1	71.7	-
MobileViTv3-0.5	1.4	1.1	74.0	21.4
MobileViT_S	5.6	1.1	78.4	35.7
tiny-MOAT-1	5.1	1.2	78.3	-
CeiT-T	6.4	1.2	76.4	-
DeiT-T	5.7	1.3	72.2	16.7
AutoFormer-tiny	5.7	1.3	74.7	25.1
EfficientFormer-L1	12.3	1.3	79.2	28.0
LocalViT-T	5.9	1.3	74.8	-
GLIT	7.2	1.4	76.3	-
CrossViT-T	6.9	1.6	73.4	19.5
MobileViTv3-S	5.1	1.9	79.3	33.2
ConvMixer-1024/12	14.6	-	77.8	35.7
PoolFormer-s12	12.0	2.0	77.2	59.0
PVT	13.1	2.1	78.7	54.5
ResNet50	25.5	4.1	78.5	29.4
DeiT-S	22.5	4.5	79.8	41.0
DenseNet-169	14.0	6.7	76.2	-
AutoViT_XXS	1.8	0.3	71.3	10.2
AutoViT_XS	2.5	0.8	75.5	19.3
AutoViT_S	6.3	1.3	79.2	27.9

71.3%, which is higher than all the other CNNs and MobileViT_XXS. Moreover, its latency of 10.2ms is the lowest among the comparable models, thus demonstrating a significant efficiency improvement. Our AutoViT_XS (19.3 ms, 75.5% Top-1 accuracy, 0.8 GFlops) and AutoViT_S (27.9 ms, 79.2% Top-1 accuracy, 1.3 GFlops) also outperform existing ViT variants in both speed and accuracy. Hand-crafted designs tend to yield a set of models that primarily differ in their dimensions. While they may serve as general-purpose models suitable for a broad range of devices, they may not be as finely optimized for a specific hardware profile as network-searched models. Our AutoViT_XS model has 36% fewer parameters (2.5M vs. 3.4M) compared to tiny-MOAT-0m with the same accuracy of 75.5%.

We measure the latency of various models on a mobile platform, displayed in Fig. 5. Models include MobileNetV1, DeiT-S, LeViT-128S, EfficientNet-B0, MobileViT_XS, as well as our own model AutoViT_XS & AutoViT_XXS. Dif-

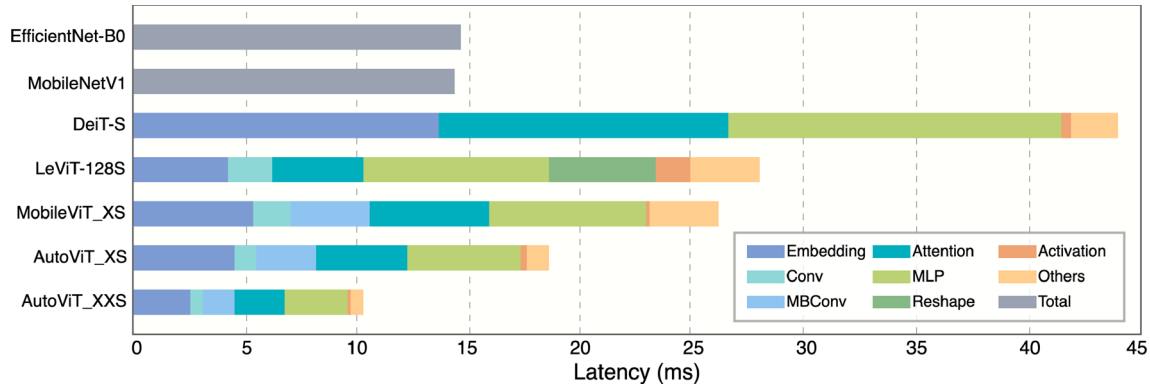

Fig. 5 Latency Breakdown. Results are obtained on iPhone 13 with CoreML. The on-device speed for various operators is reported. Models and operations are denoted with different colors

Table 3 Results comparison on COCO2017 object detection and instance segmentation

Model	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet18	34	54	36.7	31.2	51	32.7
PoolFormer-S12	37.3	59.0	40.1	34.6	55.8	36.9
EfficientFormer-L1	37.9	60.3	41.0	35.4	57.3	37.3
PVT-Tiny	36.7	59.2	39.3	35.1	56.7	37.3
AutoViT_S	37.6	60.2	41.1	35.3	57.5	37.6

Table 4 Generalization to other datasets.

Model	CIFAR10	CIFAR100	Flowers	Cars
EfficientNet-B0	98.1	88.1	96.9	90.8
DeiT-S	98.0	87.1	97.8	-
CeiT-T	98.5	88.4	96.9	90.5
CeiT-T↑384	98.5	88.0	97.8	93.0
AutoViT_S	98.8	89.6	98.1	92.4

ferent colored bars represent different modules. Notably, our searched model, particularly in the areas of attention and MLP, showcases a significant efficiency improvement. To showcase the generalizability of our method, we evaluate our model on object detection and instance segmentation benchmark, as presented in Tab. 3. Experiments are conducted on COCO 2017 (Lin et al., 2014). We use the Mask R-CNN (He et al., 2017) framework and replace different backbones. FLOPs are measured at 1333 × 800 resolution. We also conduct experiments on downstream benchmarks: CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), Flowers (Nilsback & Zisserman, 2008), Cars (Krause et al., 2013). Results are presented in Tab. 4. AutoViT_S demonstrates competitive or superior performance across different tasks and datasets. We achieved an accuracy of 89.6% in CIFAR100, which is 1.2% higher than CeiT-T.

4.3 Searched Architectures

In Tab. 5, we show our searched architectures of different size, we derive several key insights:

- Expansion ratios grow faster in early layers, favoring wider feature representations, while later layers prioritize depth scaling. This contrasts with standard ViT scaling, where depth increases uniformly. For mobile-friendly ViTs, early layers should emphasize width over depth to reduce redundancy in hierarchical feature learning.
- MBConv layers remain active even in larger models with deeper Transformer stacks. Their depth scales more slowly than Transformers, indicating that convolutional processing remains valuable for early feature extraction at all model sizes.
- Attention heads are dynamically adjusted based on stage depth and width, rather than following a uniform power-of-2 rule. Early layers benefit from fewer heads for local processing, while deeper layers require more heads for global feature fusion. This aligns with Hierarchical ViTs (e.g., Swin, CvT) but emerges automatically through NAS.

4.4 latency Analysis

The effectiveness of the proposed algorithm relies on the lookup table, correlating network blocks with latency. To ensure the accuracy of these latency values, we conducted an evaluation involving 50 randomly selected subnets. We assessed the latency of these candidates using three approaches: real latency, predictor (Wang et al., 2020), and our lookup table. In Fig. 6, for each sample, the latency aligns closely with the ground truth and is compatible with the trained predictor.

Table 5 Our searched hybrid architectures.

Block	Size	AutoViT_XXS	AutoViT_XS	AutoViT_S
Conv	Width	16	16	16
	Depth	1	1	1
	Kernel	3	3	33
MBConv-1	Width	16	16	16
	Depth	1	1	2
	Kernel	3	3	3
	Expan.	1	1	1
MBConv-2	Width	24	24	24
	Depth	2	2	3
	Kernel	3	3	3
	Expan.	2	2	2
MBConv-3	Width	32	32	32
	Depth	2	3	3
	Kernel	3	3	3
	Expan.	2	2	2
Transformer-1	Width	48	96	128
	Depth	4	4	5
	Heads	3	4	9
	Expan.	1	1.5	1.5
Transformer-2	Width	64	120	176
	Depth	3	3	4
	Heads	3	6	8
	Expan.	1.5	2	2
Transformer-3	Width	80	144	224
	Depth	3	4	4
	Heads	3	4	8
	Expan.	2	2	2
Transformer-4	Width	96	168	272
	Depth	2	4	5
	Heads	4	4	9
	Expan.	2	2	1.5
MBPool	Width	1000	c: 1280	c: 1560

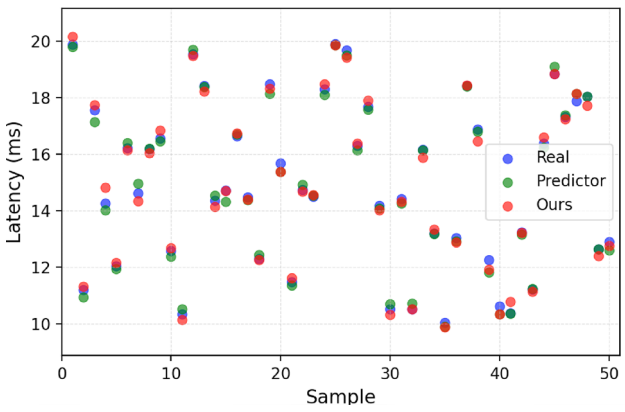

Fig. 6 Latency comparison of different latency prediction methods using 50 randomly selected subnets

4.5 Comparison with Different Search Methods

Tab. 6 shows a comparison of our method to several NAS methods, namely, ProxylessNAS (Cai et al., 2018), GM-NAS (Hu et al., 2022), OFA (Cai et al., 2019), MNasNet (Tan et al., 2019), Few-Shot NAS (Zhao et al., 2021), BigNAS (Yu et al., 2020), AlphaNet (Wang et al., 2021a), NASViT (Anonymous, 2022). One of our contributions lies in optimizing on-device latency rather than FLOPs, as FLOPs (or MACs) do not directly reflect real hardware latency across different devices. While some NAS methods report higher FLOP-based efficiency, our approach ensures better real-world speed, as demonstrated in Fig. 7, where we compare actual latency results. Additionally, AutoViT achieves superior search efficiency, reducing computational costs compared

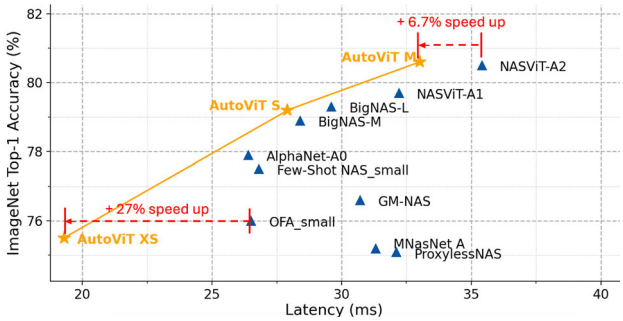

Fig. 7 Latency and accuracy comparison with different neural architecture search methods

to existing NAS methods, as shown in the right column of Tab. 6. For a fair comparison with NASViT, we conducted a latency-aware search scheme as mentioned in Section 3.5 of the paper from the pretrained AutoViT_S supernet. The only difference is that the latency threshold is set higher (35ms) for a slightly larger variant. This achieves a 6.7% speedup over NASViT-A2.

5 Ablation Study

5.1 Knowledge Distillation

In this experiment, we use knowledge distillation to improve the accuracy of the supernet. We use the pre-trained EfficientNet-B5 with 83.3% top-1 accuracy, and RegNetY-32G with 83.6% top-1 accuracy as different teacher models. We also apply soft distillation and hard distillation for comparison. Soft distillation (Hinton et al., 2015) minimizes the Kullback-Leibler divergence between the softmax of the teacher and the softmax of the student model, encouraging the student to learn richer information from the teacher's output distribution. The distillation objective is:

$$\begin{aligned}\mathcal{L}_{soft} =&(1-\alpha)\mathcal{L}_{CE}(\psi(Z_s), y)\\ &+\alpha\tau^2\mathcal{L}_{KL}(\psi(\frac{Z_s}{\tau}), \psi(\frac{Z_t}{\tau})),\end{aligned} \tag{10}$$

where Z_t and Z_s are the logits of the teacher and student model, respectively. ψ the softmax function. τ is the temperature for the distillation, and α is the coefficient balancing the KL divergence loss and the cross-entropy on ground truth labels y.

For hard-label distillation (Touvron et al., 2021), we take the hard decision of the teacher as a true label. The objective is:

$$\mathcal{L}_{hard} = (1-\alpha)\mathcal{L}_{CE}(\psi(Z_s), y) + \alpha\mathcal{L}_{CE}(\psi(Z_s), y_t), \tag{11}$$

Table 7 Distillation results with Supernet_XXS

Model	Params (M)	Top-1 Acc. (%)
w/o KD	1.8	64.6
Soft KD EffNet	1.8	68.6
Soft KD RegNet	1.8	70.5
Hard KD RegNet	1.8	**71.3**

where $y_t = argmax_c Z_t(c)$ is the hard label from the teacher logits Z_t. The results are shown in Tab. 7.

When using soft distillation, both EfficientNet-B5 and RegNetY-32G as teachers outperform the model trained without distillation (68.6% vs. 63.5% and 71.3% vs 64.6%). Moreover, we observe that although the accuracy of the RegNetY-32G model is only 0.3% higher than that of the Efficient-B5 model, the accuracy of the supernet model with RegNetY-32G as teacher outperforms the one with Efficient-B5 as a teacher by 2.7%. We also compare hard and soft distillation with RegNetY-32G as a teacher, where hard distillation outperforms soft one by 0.4%. Fig. 8 shows the loss and accuracy convergence curves of the three training methods.

5.2 Varying Number of Supernets

As discussed in Sec. 3.2, we partition the large search space into three independent sub-spaces based on parameters and train one supernet for each sub-space. This reduces gradient conflicts between large and small sub-networks trained via weight-sharing due to model size gaps. Further, search space partitioning reduces computational and memory costs. To demonstrate the benefit of our partitioning strategy for supernet training, we perform an experiment with a single supernet covering our hybrid search space. Compared to our AutoViT_XXS (71.3%) model with similar parameters, the accuracy of the searched model from the above single supernet is 6.0% lower.

Our method groups architectures based on hardware feasibility, ensuring each partition covers a meaningful range of model sizes. Additionally, overlapping parameter sizes between partitions help maintain continuity and improve search efficiency.

6 Conclusion

In this work, we apply neural architecture search to automatically search for optimal lightweight vision transformer (VIT) models given different resource constraints. We design a hybrid search space including CNN and transformer components. We train multiple supernets of different parameter

Table 6 Performance Comparison of Various Models

Model	Params (M)	FLOPS (M)	Top-1 Acc. (%)	Latency (ms)	Search Time(h)
ProxylessNAS	7.12	465	75.1	32.1	200
GM-NAS	5.2	583	76.6	30.7	26
OFA_small	4.4	230	76.0	26.5	40
MNasNet	3.9	315	75.2	31.3	$> e^5$
Few-Shot NAS-S	5.6	238	77.5	26.8	68
BigNAS-M	5.5	418	78.9	27.4	-
BigNAS-L	6.4	586	79.5	28.6	-
AlphaNet-A0	-	203	77.9	25.4	32
NASViT-A1	-	309	79.7	32.2	35
NASViT-A2	-	421	80.5	35.4	35
AutoViT_XS	2.5	804	75.5	19.3	12
AutoViT_S	6.3	1318	79.2	27.9	12
AutoViT_M	6.4	1426	80.6	33.0	12

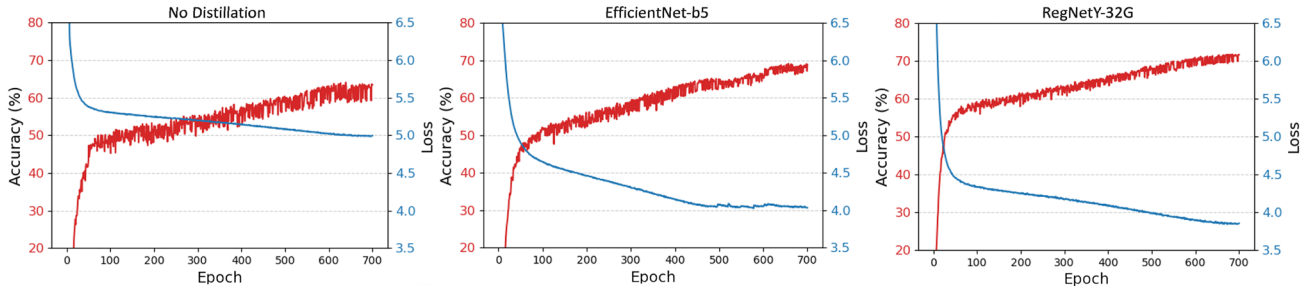

Fig. 8 We compare the loss and accuracy convergence curves of the three training methods for Supernet _XXS. From left to right: no distillation, distillation with EfficientNet-B5 as a teacher, and distillation with RegNetY-32G as a teacher. Models trained with RegNetY-32G as the teacher outperform the other model trained with Efficient-B5 and those without distillation

ranges to handle the huge search space in ViT and mitigate conflicts in the weight-sharing of sub-networks. Furthermore, we introduce an efficient latency-aware coarse-to-fine search algorithm to obtain the optimal networks that significantly outperform prior-art lightweight vision transformer models. Our approach leads to improved efficiency compared to testing the speed of the whole model during the search process.

Data Availability In this work, all our datasets are publicly available: ImageNet-1K (Deng et al., 2009), which spans 1000 object classes and contains 1,281,167 training images, 50,000 validation images, and 100,000 test images. This subset is available on https://www.image-net.org/download.php. COCO (Common Objects in Context) (Lin et al., 2014) is a large-scale object detection, segmentation, and captioning dataset. It contains over 200,000 labeled images with over 80 category labels. This dataset is available on https://cocodataset.org/#home. The CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) are labeled subsets of the 80 million tiny images dataset. They are available to download at https://www.cs.toronto.edu/\protect\unhbox\voidb@x\penalty\@M\kriz/cifar.html. Oxford 102 Flower (Nilsback & Zisserman, 2008) is an image classification dataset consisting of 102 flower categories. The flowers chosen are commonly occurring in the United Kingdom. Each class consists of between 40 and 258 images. This dataset is available on https://www.robots.ox.ac.uk/\protect\unhbox\voidb@x\penalty\@M\vgg/data/flowers/102/. The Cars (Krause et al., 2013) dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images. This dataset is available on https://www.kaggle.com/datasets/jessicali9530/stanford-cars-dataset.

References

Anonymous. (2022). NASVit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training. In *Submitted to The Tenth International Conference on Learning Representations*, under review.

Bender, Gabriel, Liu, Hanxiao, Chen, Bo, Chu, Grace, Cheng, Shuyang, Kindermans, Pieter-Jan, Le, Quoc V. (2020). Can weight sharing outperform random architecture search? an investigation with tunas. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14323–14332

Cai, Han, Gan, Chuang, Wang, Tianzhe, Zhang, Zhekai, & Han, Song. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791

Cai, Han, Zhu, Ligeng, & Han, Song. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332

Chen, Boyu, Li, Peixia, Li, Chuming, Li, Baopu, Bai, Lei, Lin, Chen, Sun, Ming, Yan, Junjie, & Ouyang, Wanli. (2021). Glit: Neural architecture search for global and local image transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–21

Chen, Chun-Fu, Fan, Quanfu, & Panda, Rameswar. (2021). Crossvit: Cross-attention multi-scale vision transformer for image classification. arXiv preprint arXiv:2103.14899

Chen, Hanting, Wang, Yunhe, Guo, Tianyu, Xu, Chang, Deng, Yiping, Liu, Zhenhua, Ma, Siwei, Xu, Chunjing, Xu, Chao, & Gao, Wen. (2021). Pre-trained image processing transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12299–12310

Chen, Minghao, Peng, Houwen, Fu, Jianlong, & Ling, Haibin. (2021). Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12270–12280

Chen, Minghao, Wu, Kan, Ni, Bolin, Peng, Houwen, Liu, Bei, Fu, Jianlong, Chao, Hongyang, & Ling, Haibin. (2021). Searching the search space of vision transformer. *Advances in Neural Information Processing Systems*, 34

Chen, Tianlong, Cheng, Yu, Gan, Zhe, Yuan, Lu, Zhang, Lei, & Wang, Zhangyang. (2021). Chasing sparsity in vision transformers: An end-to-end exploration. In *Advances in Neural Information Processing Systems*

Cheng, Bowen, Schwing, Alexander G., & Kirillov, Alexander. (2021). Per-pixel classification is not all you need for semantic segmentation. arXiv preprint arXiv:2107.06278

Chu, Xiangxiang, Tian, Zhi, Zhang, Bo, Wang, Xinlong, Wei, Xiaolin, Xia, Huaxia, & Shen, Chunhua. (2021). Conditional positional encodings for vision transformers. arXiv preprint arXiv:2102.10882,

Cubuk, Ekin D., Zoph, Barret, Shlens, Jonathon, & Le, Quoc V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703

Dai, Zhigang, Cai, Bolun, Lin, Yugeng, & Chen, Junying. (2021). Up-detr: Unsupervised pre-training for object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1601–1610

Dai, Zihang, Liu, Hanxiao, Le, Quoc V., & Tan, Mingxing. (2021). Coatnet: Marrying convolution and attention for all data sizes. arXiv preprint arXiv:2106.04803

d'Ascoli, Stéphane, Touvron, Hugo, Leavitt, Matthew, Morcos, Ari, Biroli, Giulio, & Sagun, Levent. (2021). Convit: Improving vision transformers with soft convolutional inductive biases. arXiv preprint arXiv:2103.10697

Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, & Fei-Fei, Li. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee

Dong, P., Kong, Z., Meng, X., Yu, P., Gong, Y., Yuan, G., Tang, H., & Wang, Y. (2023a). HotBEV: Hardware-oriented transformer-based multi-view 3D detector for BEV perception. *Advances in Neural Information Processing Systems, 36*, 2824–2836.

Dong, P., Kong, Z., Meng, X., Zhang, P., Tang, H., Wang, Y., & Chou, C. H. (2023b). SpeedDETR: Speed-aware transformers for end-to-end object detection. In *Proceedings of the 40th International Conference on Machine Learning*. JMLR.org.

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

Graham, Benjamin, El-Nouby, Alaaeldin, Touvron, Hugo, Stock, Pierre, Joulin, Armand, Jegou, Herve, & Douze, Matthijs. (2021). Levit: A vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12259–12269, October

Guo, Zichao, Zhang, Xiangyu, Mu, Haoyuan, Heng, Wen, Liu, Zechun, Wei, Yichen, & Sun, Jian. (2020). Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer

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

Heo, Byeongho, Yun, Sangdoo, Han, Dongyoon, Chun, Sanghyuk, Choe, Junsuk, & Oh, Seong Joon. (2021). Rethinking spatial dimensions of vision transformers. In *International Conference on Computer Vision (ICCV)*

Hinton, Geoffrey, Vinyals, Oriol, & Dean, Jeff. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531

Howard, Andrew G., Zhu, Menglong, Chen, Bo, Kalenichenko, Dmitry, Wang, Weijun, Weyand, Tobias, Andreetto, Marco, & Adam, Hartwig. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861

Hu, Shoukang, Wang, Ruochen, Hong, Lanqing, Li, Zhenguo, Hsieh, Cho-Jui, & Feng, Jiashi. (2022). Generalizing few-shot nas with gradient matching. arXiv preprint arXiv:2203.15207

Huang, Gao, Liu, Zhuang, Der Maaten, Laurens Van, & Weinberger, Kilian Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708

Krause, Jonathan, Stark, Michael, Deng, Jia, & Fei-Fei, Li. (2013). 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561

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

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

Li, Yawei, Zhang, Kai, Cao, Jiezhang, & Timofte, Radu. (2021). *and Luc Van Gool*. Bringing locality to vision transformers: Localvit.

Li, Z., Yuan, G., Niu, W., Zhao, P., Li, Y., Cai, Y., Shen, X., Zhan, Z., Kong, Z., Jin, Q. and Chen, Z., Liu, S., Yang, K., Ren, B., Wang, Y., & Lin, X. (2021). Npas: A compiler-aware framework of unified network pruning and architecture search for beyond real-time

mobile acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 14255-14266).

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

Liu, Ze, Lin, Yutong, Cao, Yue, Hu, Han, Wei, Yixuan, Zhang, Zheng, Lin, Stephen, & Guo, Baining. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*

Lu, Zhisheng, Liu, Hong, Li, Juncheng, & Zhang, Linlin. (2021). Efficient transformer for single image super-resolution. arXiv preprint arXiv:2108.11084

Ma, Ningning, Zhang, Xiangyu, Zheng, Hai-Tao, & Sun, Jian. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September

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

Mehta, Sachin, Rastegari, Mohammad, Shapiro, Linda, & Hajishirzi, Hannaneh. (2019). Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9190–9200

Misra, Ishan, Girdhar, Rohit, & Joulin, Armand. (2021). An End-to-End Transformer Model for 3D Object Detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*

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

Pham, Hieu, Guan, Melody, Zoph, Barret, Le, Quoc, & Dean, Jeff. (2018). Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR

Poyser, Matt, & Breckon, Toby P. (2024). Neural architecture search: A contemporary literature review for computer vision applications. *Pattern Recognition, 147*, Article 110052.

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

Tan, Mingxing, Chen, Bo, Pang, Ruoming, Vasudevan, Vijay, Sandler, Mark, Howard, Andrew, & Le, Quoc V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828

Tan, Mingxing, & Le, Quoc. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR

Touvron, Hugo, Cord, Matthieu, Douze, Matthijs, Massa, Francisco, Sablayrolles, Alexandre, & J'egou,Herv'e. (2021). Training data-efficient image transformers & distillation through attention. In *ICML*

Wan, Xingchen, Ru, Binxin, Esparança, Pedro M., & Carlucci, Fabio Maria. (2022). Approximate neural architecture search via operation distribution learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2377–2386

Wang, Dilin, Gong, Chengyue, Li, Meng, Liu, Qiang, & Chandra, Vikas. (2021). Alphanet: Improved training of supernet with alpha-divergence. arXiv preprint arXiv:2102.07954

Wang, Dilin, Li, Meng, Gong, Chengyue, & Chandra, Vikas. (2021). Attentivenas: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6427

Wang, Hanrui, Wu, Zhanghao, Liu, Zhijian, Cai, Han, Zhu, Ligeng, Gan, Chuang, & Han, Song. (2020). Hat: Hardware-aware transformers for efficient natural language processing. arXiv preprint arXiv:2005.14187

Wang, Wenhai, Xie, Enze, Li, Xiang, Fan, Deng-Ping, Song, Kaitao, Liang, Ding, Lu, Tong, Luo, Ping, & Shao, Ling. (2021). Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)

Xia, Xin, Xiao, Xuefeng, Wang, Xing, & Zheng, Min. (2022). Progressive automatic design of search space for one-shot neural architecture search. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 2455–2464

Yang, Chenglin, Qiao, Siyuan, Yu, Qihang, Yuan, Xiaoding, Zhu, Yukun, Yuille, Alan, Adam, Hartwig, & Chen. Liang-Chieh. (2022). Moat: Alternating mobile convolution and attention brings strong vision models. arXiv preprint arXiv:2210.01820

Yu, Jiahui, Jin, Pengchong, Liu, Hanxiao, Bender, Gabriel, Kindermans, Pieter-Jan, Tan, Mingxing, Huang, Thomas, Song, Xiaodan, Pang, Ruoming, & Le, Quoc. (2020). Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pages 702–717. Springer

Yu, Weihao, Luo, Mi, Zhou, Pan, Si, Chenyang, Zhou, Yichen, Wang, Xinchao, Feng, Jiashi, & Yan, Shuicheng. (2022). Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10819–10829

Yuan, Kun, Guo, Shaopeng, Liu, Ziwei, Zhou, Aojun, Yu, Fengwei, & Wu, Wei.(2021). Incorporating convolution designs into visual transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 579–588, October

Yuan, Li, Chen, Yunpeng, Wang, Tao, Yu, Weihao, Shi, Yujun, Jiang, Zi-Hang, Tay, Francis E.H., Feng, Jiashi, & Yan, Shuicheng. (2021). Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 558–567, October

Yun, Sangdoo, Han, Dongyoon, Oh, Seong Joon, Chun, Sanghyuk, Choe, Junsuk, & Yoo, Youngjoon. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032

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

Zhao, Yiyang, Wang, Linnan, Tian, Yuandong, Fonseca, Rodrigo, & Guo, Tian. (2021). Few-shot neural architecture search. In *International Conference on Machine Learning*, pages 12707–12718. PMLR

Zheng, Sixiao, Lu, Jiachen, Zhao, Hengshuang, Zhu, Xiatian, Luo, Zekun, Wang, Yabiao, Fu, Yanwei, Feng, Jianfeng, Xiang, Tao, Torr, Philip H.S. et al. (2021). Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6881–6890

Xin, Y., Luo, S., Liu, X., Du, Y., Zhou, H., Cheng, X., Lee, C. E., Du, J., Wang, H., Chen, M., Liu, T., Hu, G., Wan, Z., Zhang, R., Li, A., Yi, M., Liu, X. (2024a). V-petl bench: A unified visual parameter-efficient transfer learning benchmark. *Advances in Neural Information Processing Systems, 37*, 80522–80535.

Xin, Y., Du, J., Wang, Q., Lin, Z., Yan, K. (2024b). Vmt-adapter: Parameter-efficient transfer learning for multi-task dense scene

understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 38, No. 14, pp. 16085–16093).

Xin, Y., Luo, S., Zhou, H., Du, J., Liu, X., Fan, Y., Li, Q., & Du, Y. (2024c). Parameter-efficient fine-tuning for pre-trained vision models: A survey. arXiv:2402.02242

Xin, Y., Du, J., Wang, Q., Yan, K., & Ding, S. (2024d). Mmap: Multi-modal alignment prompt for cross-domain multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 38, No. 14, pp. 16076–16084).

Xin, Y., Luo, S., Jin, P., Du, Y., & Wang, C. (2023). Self-training with label-feature-consistency for domain adaptation. In *International Conference on Database Systems for Advanced Applications* (pp. 84–99). Springer Nature Switzerland.

Zhong, Zhun, Zheng, Liang, Kang, Guoliang, Li, Shaozi, & Yang, Yi. (2020). Random erasing data augmentation. *In Proceedings of the AAAI Conference on Artificial Intelligence, 34*, 13001-13008.

Yu, P., Kong, Z., Zhao, P., Dong, P., Tang, H., Sun, F., Lin, X., & Wang, Y. (2025, February). Q-TempFusion: Quantization-Aware Temporal Multi-Sensor Fusion on Bird's-Eye View Representation. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (pp. 5489–5499). IEEE.