Lightweight Vision Transformer Course-to-Fine Search via Latency Profiling

Anonymous authors
Paper under double-blind review

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. This requires re-running the training process to obtain different lightweight ViTs under different resource constraints. In this work, we address the challenge of finding optimal lightweight ViTs given constraints on model size and computational cost using neural architecture search. Using the proposed method, we first train a supernet, which is a hybrid architecture of CNNs and transformers. To efficiently search for the optimal architecture, 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. With extensive experiments on ImageNet, we demonstrate that under similar parameters and FLOPs, our searched lightweight ViTs have higher accuracy and lower latency compared to state-of-the-art models. For example, our AutoViT_XXS (71.3% Top-1 accuracy and 10.2ms latency) has a 2.3% higher accuracy and 4.5ms lower latency compared to MobileViT_XXS (69.0% Top-1 accuracy of and 14.7ms latency).

1 Introduction

Vision Transformer (ViT) Dosovitskiy et al. (2020) exploits the self-attention mechanism inherited from the transformer architecture and has recently obtained state-of-the-art performance in vision tasks like image classification Dosovitskiy et al. (2021); Touvron et al. (2021); Chen et al. (2021b), object detection Carion et al. (2020); Dai et al. (2021a); Amini et al. (2021); Misra et al. (2021); Dai et al. (2022), semantic segmentation Zheng et al. (2021); Cheng et al. (2021); Ding et al. (2021), image retrieval El-Nouby et al. (2021), image enhancement Yang et al. (2020); Chen et al. (2021c); Lu et al. (2021). These efforts have focused on improving the overall performance of the vision transformers. While the overall results are impressive, ViTs sacrifice lightweight model capacity and portability for high accuracy. Vision transformers are relatively less explored for mobile vision tasks Mehta & Rastegari (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 ViTs that have emerged recently d'Ascoli et al. (2021); Chu et al. (2021); Dai et al. (2021b); Liu et al. (2021); Graham et al. (2021); Wang et al. (2021b). Our objective is to automate the search process for hybrid vision transformers under variable resource constraints.

Neural architecture search (NAS) has shown its benefit over manual design as a powerful technique for the automatic design of neural networks. The recently proposed BigNAS Yu et al. (2020) trains a supernet that covers all subnets in the architecture search space with weight sharing. Sub-models of various sizes can be generated directly without additional training to obtain optimal results. However, applying NAS for vision transformers has the following challenges: (i)

While researchers have invested significant effort in designing the search space for CNNs Sahni et al. (2021), there has been limited exploration on design choices for transformer-based models, especially vision transformers. (ii) During the supernet training process, one of the most popular paradigms for NAS, large subnets and small subnets encounter interference from gradient conflicts, making the subnets converge poorly Yu et al. (2020); Chen et al. (2021d); Anonymous (2022). (iii) Due to the large search space, typical search algorithms (e.g., evolutionary algorithms) lead to large training costs.

There are some recent works on using neural architecture search for automated ViT design with some limitations. For instance, Autoformer Chen et al. (2021d) entangles the weights of different visual transformer blocks in the same layer during supernet training and trains three different scales of supernets to reduce the training time. However, their search space is limited to the original transformer-only architecture leading to performance degradation for very lightweight models. 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. In this work, we explore the challenge of *designing an efficient and accurate algorithm to search for the optimal structure regarding latency and accuracy trade-off for memory & computation-restricted devices.*

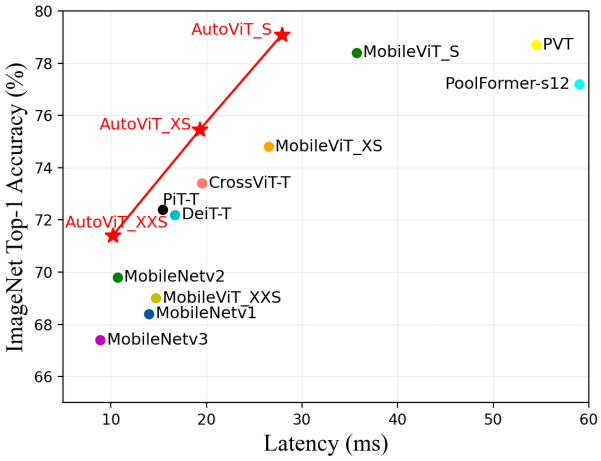

Figure 1: Top-1 accuracy and sizes of different models on ImageNet. Our method achieves a better trade-off than other efficient CNN-based and transformer-based models.

We aim to develop a series of lightweight ViTs with a computational cost of under 1 GFLOP. 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 Figure 1. To improve training efficiency, we encode the search space into independent supernets, which reduces optimization interference between subnets of different sizes. Our experiments show that this approach leads to improved performance compared to existing methods.

In the field of edge computing, the selection of models pivots critically on the metric of latency. Despite this, numerous prevailing search algorithms face challenges in accurately evaluating model latency using only FLOPs and parameter counts. As such, a model that appears smaller in size as a result of a search process does not necessarily guarantee higher speed. In light of these limitations, some researchers have made attempts to incorporate latency into the search criteria via the utilization of a latency predictor. However, developing an accurate predictor is not without its 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 technique 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:

- **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 address the challenges of dealing with a large search space in vision transformers and optimize the subnets of different sizes, we proposed a multi-size supernet training scheme that mitigates optimization interference between subnets and enables 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 among the trained supernets.

- **Performence:** Our searched model can achieve up to 79.2% accuracy on ImageNet with 6.3M parameters and 1.4 GFLOPs, exceeding the performance of existing models with similar resource budgets.

2 Related Work

Vision Transformers. ViT Dosovitskiy et al. (2020) 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), and Mixer Tolstikhin et al. (2021) 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 input token and attention head sparsity. Recent works have also attempted to design various self-supervised learning schemes for vision transformers to address the data-hungry issue. BEiT Bao et al. (2021) and PeCo Dong et al. (2021) pre-train ViT by masked image modeling with an image tokenizer based on dVAE that vectorizes images into discrete visual tokens. MAE He et al. (2021) eliminates the dVAE pre-training process by reconstructing pixels, in contrast to predicting tokens. MaskFeat Wei et al. (2021) also replaces dVAE visual tokens with HOG prediction.

Neural Architecture Search. There has been increasing interest in designing efficient networks with neural architecture search (NAS). Among different methods, weight-sharing NAS has become popular due to training efficiency Yu et al. (2020); Wang et al. (2021a); Sahni et al. (2021). They train one over-parameterized supernet whose weights are shared across all sub-networks in the search space to conduct architecture search, significantly reducing the computational cost. However, many of these works focus on CNN-only architectures with well-designed search space Sahni et al. (2021). Some recent works search for efficient and optimal transformer-based architecture. HAT Wang et al. (2020) searches for an encoder-decoder transformer structure and requires additional retraining or finetuning of the optimal candidates obtained during the search. BossNAS Li et al. (2021a) searches for CNN-transformer hybrid models with block-wisely self-supervised neural architecture search. CvT Wu et al. (2021) proposes a new architecture family and searches for strides and kernel size. Autoformer Chen et al. (2021d) entangles the weights of different vision transformer blocks in the same layer during supernet training and trains three supernets with different scales to reduce training time. S3 Chen et al. (2021e) adds search space search before training supernet to optimize the search space. However, these works only explore transformer structure. GLiT Chen et al. (2021a) searches for a hybrid architecture with the best self-attention and convolution head ratio. NASVIT Anonymous (2022) extends the search space to get efficient models leveraging a hybrid architecture of convolutions and transformers. However, these works incur extensive training and search cost due to the extended search space.

Our advantages over existing works can be summarized as follows: ① In contrast to hand-crafted architectures such as MobileViT Mehta & Rastegari (2021), we leverage NAS to automatically find optimal model architectures on the pareto-optimal frontier of accuracy and FLOPs. ② Most existing NAS methods operate on CNN-only (e.g., BigNAS Yu et al. (2020), MobileNet Howard et al. (2017)) or Transformer-only (e.g., AutoFormer Chen et al. (2021d), AutoDistill Xu et al. (2022)) search spaces. We design a better search space that combines the advantages of CNN (e.g., spatially induced bias) and ViT (e.g., input adaptive weighting and global processing) to search for lightweight and high-performance hybrid models. ③ NASVIT Anonymous (2022) searches for hybrid ViTs, but their extended search space significantly increases the training cost (4 days on 80 V100 GPUs). We apply a more efficient training and searching strategy and reach comparable performance. ④ Opposing the one-shot block-wisely search scheme of BossNAS Li et al. (2021a), we leverage a more flexible weight-sharing method and train with three supernets of different sizes individually to diminish the co-adaptation issue.

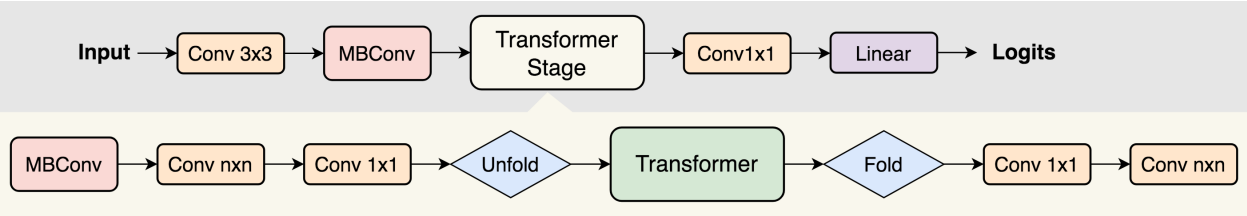

Figure 2: An illustration of our ViT structure. Conv and MBConv refer to standard convolution and inverted residual blocks, respectively. All CNN and transformer blocks contain a stack of dynamic layers with searchable architecture configurations. We also search for the input resolutions.

3 Methodology

3.1 Background

Our goal is 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. For example, Swin transformer Liu et al. (2021) achieves SOTA accuracy on multiple computer vision tasks. In the field of object detection, it is only applied to frameworks such as Retina Lin et al. (2017) or Maskrcnn He et al. (2017), but has not yet been applied to frameworks such as YOLO series Redmon et al. (2016); Redmon & Farhadi (2017; 2018); Bochkovskiy et al. (2020) that are known for their efficiency. (ii) Most of the current works 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 when compared to CNN architectures that are 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 of these approaches is that they are spatially localized. On the other hand, the transformer is able to 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 of convolution and transformer that considers both performance and efficiency. Past work Mehta & Rastegari (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 Figure 2, each of our hybrid blocks consists of standard convolution layers and one transformer block. Transformer stands for the transformer blocks and MBConv refers to MobileNetv2 Sandler et al. (2018) blocks. The standard convolution involves three main operations: unfolding, local processing, and folding. The transformer block is mainly used to obtain global modeling dependencies. The combination of these two approaches can effectively encode local and global information. Following Mehta & Rastegari (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 (or d-dimensional, where $d > C$) by learning linear combinations of the input channels. To learn global representations with spatial inductive bias, we unfold X_L into N non-overlapping flattened patches $X_U \in \mathbb{R}^{P \times N \times d}$, where P is the number of pixels and N is the number of patches. $h \leq n$ and $w \leq n$ are height and width of a patch, respectively. For each $p \in \{1, ..., P\}$ inter-patch relationships are encoded by applying transformers to obtain $X_G \in \mathbb{R}^{P \times N \times d}$ as $X_G(p) = Transformer(X_U(p))$,

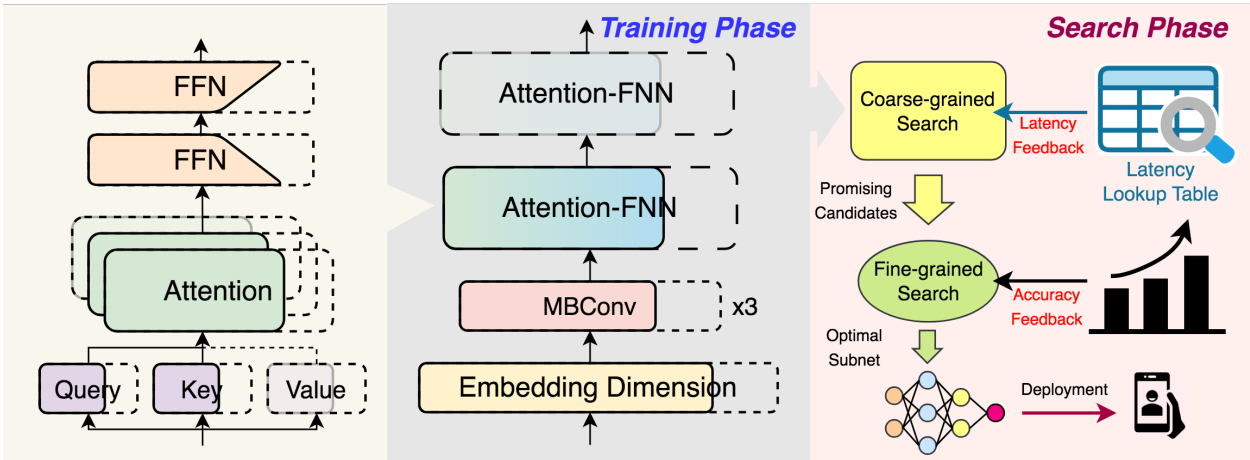

Figure 3: **Framework Overview.** We train a weight-shared supernet by iteratively optimizing randomly sampled subnets. *Left:* We search for the number of heads and expansion ratio of a transformer block. *Middle:* We search for the width and depth of MBConv and transformer block. Each layer and block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

where $1 \leq p \leq P$. This type of transformer block prevents ViT from losing either the patch order or the spatial order of pixels within each patch.

We summarize the detailed dimensions of our search space. In Table 1, width represents the channel size for CNN 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 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. It also reduces gradient conflicts between large and small sub-networks trained via weight-sharing due to gaps in model sizes.

3.3 Neural Architecture Search Pipeline

We apply one-shot NAS, which includes two phases: (i) Train a supernet containing all the candidate subnetworks. During training, the parameters of all candidate networks in the search space are optimized simultaneously by weight sharing. (ii)) Search for the best sub-network in the well-trained supernet under various resource constraints Cai et al. (2019). Typical search techniques include evolutionary algorithms Guo et al. (2020). Figure 3 shows our overall

framework. We use the search space introduced in Table 1 and train a weight-shared supernet by iteratively optimizing randomly sampled subnets from the space. We search for the width, depth, and expansion ratio of both CNN layers and transformer layers and also the number of self-attention heads. The layer and depth in each block are dynamic. 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, which can efficiently estimate the latency of network candidates by considering network properties, hardware memory access pattern, and degree of parallelism.

3.3.1 Supernet Training

In transformer search space, the classical supernet training strategy encounters the following challenges. (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 Entanglement. Inspired by Autoformer Chen et al. (2021d) operating on transformer-only search space, we apply the weight entanglement training strategy to the vision transformer search space. The central idea is to enable different transformer blocks to share weights for their common parts in each layer. The weight entanglement strategy works specifically for homogeneous building blocks. This is because homogeneous blocks are structurally compatible so that weights can be shared with each other. 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.

Compared with classical weight-sharing methods, the weight entanglement strategy has three advantages. (i) Faster convergence. Weight entanglement allows each block to be updated more times than the previous independent training strategy. (ii) Low memory cost. We now only need to store the largest building blocks' parameters for each layer, instead of all the candidates in the space. (iii) Better performance of the subnets.

Table 1: An illustration of our search space: The search space is divided into three individual supernets within different parameter ranges to satisfy different resource constraints. Tuples of three values in parentheses represent the lowest value, highest, and steps.

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}	-	-	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}	-	-	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	{1000}	-	-	6
Number of Stage		{3, 4}		
Params Range		$4.0 \sim 7.2$M		

We randomly select a transformer architecture in each iteration. Then we obtain its weights from the weights of the supernet and compute the losses of the subnets. Finally, we update the corresponding weights with the remaining weights frozen. The architecture search space P is encoded in a supernet, denoted as $\mathcal{S}(P, W)$, where W is the weight of the supernet that is shared across all the candidate architectures s. Algorithm 1 illustrates the training procedure of our supernet. During the training, the weight W is optimized by:

$$W = \arg\min_{W} \mathcal{L}(\mathcal{S}(P, W)), \tag{1}$$

where $\mathcal{L}$ represents the loss function on the training dataset. To reduce memory usage, one-shot methods usually sample subnets from $\mathcal{S}$ for optimization. The second stage is to search architectures by ranking the performance of subnets based on the learned weights in W. The subnet search objective is:

$$s = \arg\max_{s \in \mathcal{S}} Acc(\mathcal{S}(p, w)), \tag{2}$$

where Acc indicates the top-1 accuracy of the architecture on the validation dataset, s is the sampled subnet that inherits weight w from W.

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. (2021b) and the Swin Transformer Liu et al. (2021). (2) Weight entanglement allows different transformer blocks to share weights of their common parts in each layer with significant benefits of faster convergence, reduced memory footprint, and improved subnet performance.

Algorithm 1 Supernet Training.

Input: Training epochs N , search space $\mathcal{P}$, supernet $\mathcal{S}$, loss function L, train dataset D_{train}, initial supernet weights $\mathcal{W}_A$, candidate weights w
for i in N epochs **do**
  for data, labels in D_{train} **do**
    Randomly sample one transformer architecture from search space $\mathcal{P}$
    Obtain the corresponding weights w from $\mathcal{W}_A$
    Compute the gradients based on L
    Update the corresponding part of w in $\mathcal{W}_A$ while freezing the rest of the supernet $\mathcal{S}$
  end for
end for
Output $\mathcal{S}$

3.3.2 Latency-aware Search Scheme

Latency Profiling. Upon acquisition of the trained supernet, we carry out a search algorithm to derive the ideal subnets. Predominant 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, such as HAT, 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.

To overcome this challenge, 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}, thus making a total of 46 modules. It's noteworthy that the speed of individual module execution may differ from when they are combined due to inter-module influences. Given that we do not calculate the latency of each module in isolation, we measure the latency of each module based on the per-latency accuracy drop when removed from the entire model. This methodology provides a more comprehensive and realistic understanding of the latency impacts.

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.

Figure 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

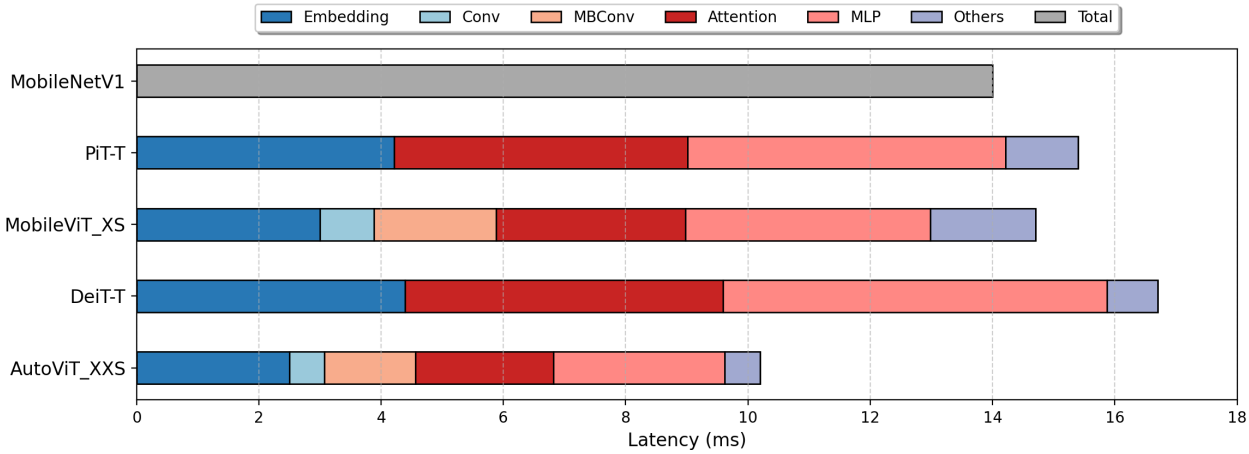

Figure 4: **Latency Breakdown.** Results are obtained on iPhone 13 with CoreML. The on-device speed for various operators is reported. The latency of models and operations are denoted with different colors.

our latency budget are selected to proceed to the fine-grained search phase. Here, we conduct an evolutionary search to procure the optimal subnets (as shown in Figure 5). Our objective is to maximize classification accuracy while minimizing the model size. 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. Algorithm 2 elaborates the procedure of our search method.

Algorithm 2 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

4 Experiments

4.1 Datasets and Implementation Details

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

Table 2: Accuracy comparison on ImageNet with state-of-the-art CNN and transformer-based models under similar parameters and computational cost (FLOPs).

Model	Params (M)	FLOPS (G)	Top-1 Accuracy (%)	Latency (ms)
MobileNetv1	2.6	0.3	68.4	14.0
MobileNetv2	2.6	0.2	69.8	10.7
MobileNetv3	2.5	0.1	67.4	8.9
ShuffleNetv2	2.3	0.1	69.4	-
ESPNetv2	2.3	0.2	69.2	-
MobileViT_XXS	1.3	0.2	69.0	14.7
AutoViT_XXS	1.8	0.3	71.3	**10.2**
DeiT-T	5.7	1.3	72.2	16.7
T2T-T	4.3	1.1	71.7	-
PiT-T	10.6	0.7	72.4	15.4
CrossViT-T	6.9	1.6	73.4	19.5
MobileViT_XS	2.3	0.6	74.8	26.5
AutoViT_XS	2.5	0.8	75.5	**19.3**
DenseNet-169	14	6.7	76.2	-
EfficientNet-B0	5.3	0.4	76.3	14.5
LocalViT-T	5.9	1.3	74.8	-
CeiT-T	6.4	1.2	76.4	-
PVT	13.1	2.1	78.7	54.5
MobileViT_S	5.6	1.1	78.4	35.7
AutoViT_S	6.3	1.3	79.2	**27.9**

Table 3: Searched hybrid architectures from the three supernets.

ViT Model	Width	Depth	Number of heads	Expantion ratio
AutoViT_XXS	[48, 64, 80, 96, 112]	[5]	[3, 3, 3, 4, 2]	[2, 1.5, 2, 2, 2]
AutoViT_XS	[96, 120, 144, 168]	[4]	[4, 6, 4, 4]	[1.5, 2, 2, 2]
AutoViT_S	[128, 176, 224, 272]	[4]	[9, 8, 8, 9]	[1.5, 2, 2, 1.5]

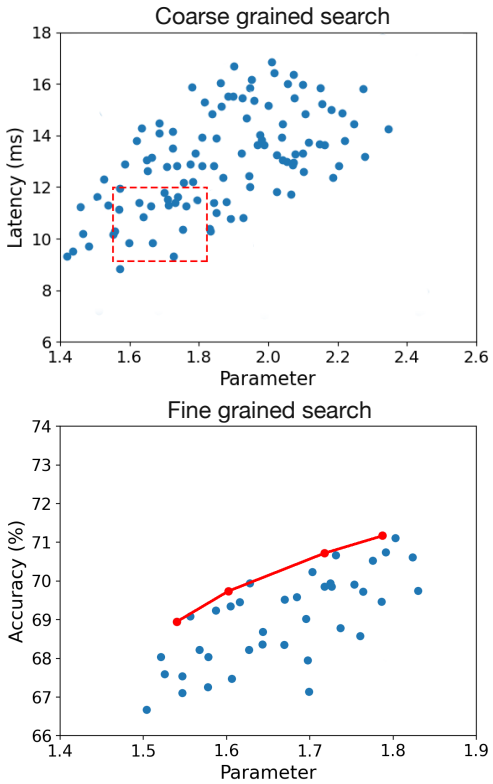

Figure 5: Coarse-to-fine Architecture Search. *Upper figure:* Pre-define some promising candidate network skeletons by the number of parameters and latency budget. *Bottom figure:* Generate more architectures by randomly mutating each skeleton architecture by varying the depth slightly and using the weights from the single-stage model for the induced child models to evaluate all the candidates.

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 Nvidia Tesla 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, respectively.

4.2 Experimental Results

We conduct an architecture search on ImageNet and obtain 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 Table 2. We compare our hybrid models with various CNN-based and Transformer based models, namely: MobileNets Howard et al. (2017), ShuffleNetv2 Ma et al. (2018), ESPNetv2 Mehta et al. (2019), MobileViT Mehta & Rastegari (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), EfficientNet-B Tan & Le (2019), LocalViT-T Li et al. (2021b), CeiT-T Yuan et al. (2021a), and PVT Wang et al. (2021b). As shown in Figure 1, our searched hybrid models obtain a better trade-off in model size vs. Top-1 accuracy in ImageNet compared to other models. The configurations of our final searched hybrid models are shown in Table 3. Our AutoViT_XXS, with only 1.8M parameters and 0.3G FLOPs, achieves a Top-1 accuracy of 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

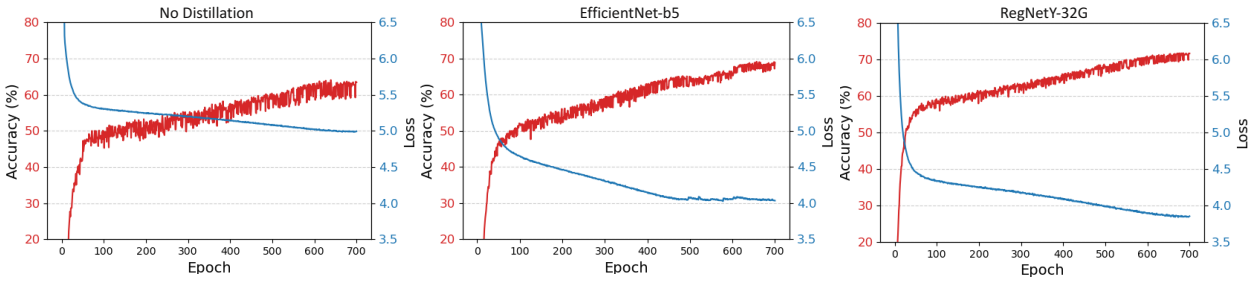

Figure 6: We compare the loss and accuracy convergence curves of the three training methods for Supernet _XXS. From left to right: no distillation, distillation with EfficientNet-B5 as a teacher, and distillation with RegNetY-32G as a teacher. Models trained with RegNetY-32G as the teacher outperform models trained with Efficient-B5 and those without distillation.

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.

We measure the latency of various models on a mobile platform, displayed in Figure 4. Models include MobileNetV1, PiT-T, MobileViT_XS, DeiT-T, and our own model AutoViT_XXS. Different colored bars represent different modules. Notably, our searched model, particularly in the areas of attention and MLP (Multilayer Perceptron), showcases a significant efficiency improvement.

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. The distillation loss is:

$$L_{soft} = (1-\alpha)L_{CE}(\psi(Z_s), y) + \alpha\tau^2 KL(\psi(\frac{Z_s}{\tau}), \psi(\frac{Z_t}{\tau})), \tag{3}$$

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, α is the coefficient balancing the Kullback–Leibler divergence loss (KL) and the cross-entropy (LCE) 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 distillation objective is:

$$L_{hard} = (1-\alpha)L_{CE}(\psi(Z_s), y) + \alpha L_{CE}(\psi(Z_s), y_t), \tag{4}$$

where $y_t = argmax_c Z_t(c)$ is the hard label from the teacher logits Z_t. The results are shown in Table 4. 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%. Figure 6 shows the loss and accuracy convergence curves of the three training methods.

5.2 Varying Number of Supernets

As discussed in Section 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 XXS model with similar parameters (1.8M), the accuracy of the searched model from the above single supernet achieved a Top-1 accuracy of 65.3%, which is 6.0% lower than the accuracy obtained with the XXS supernet (71.3%).

Table 4: Distillation results on ImageNet with Supernet_XXS. We compare the results of our supernet trained with no distillation versus EfficientNet-B5 and RegNetY-32G as teachers for soft and hard distillation.

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**

Table 5: Latency Comparison between Hybrid and Transformer-only Model.

Model	Latency (ms)	Top-1 Acc. $(\%)$
Transformer-Only	16.1	71.2
Hydbrid	10.2	71.3

Table 6: Training time (GPU hours) comparison of different search methods on Supernet_XXS.

Model	Training Time (h)	Top-1 Acc. $(\%)$
Random	0	66.8
Evolution	12	70.7
Course-to-fine	5	71.3

Table 7: Comparison with existing ViT neural architecture search baselines.

Model	Params(M)	FLOPS(G)	Top-1 Acc. $(\%)$
ASViT	29	5.3	81.2
GLiT	7.2	1.4	76.3
Autoformer	5.7	1.3	74.7
AutoViT_S	6.3	1.4	**79.2**

5.3 Hybrid Vs. Transformer-Only Structure.

Table 5 provides a comparison of latency and Top-1 Accuracy between a Transformer-Only model and a Hybrid model. It is observed under similar Top-1 Accuracy (71.2% for Transformer-Only and 71.3% for the Hybrid model), the Hybrid model's latency is significantly lower, approximately 37% less than the Transformer-Only model (10.2ms for the Hybrid model compared to 16.1ms for the Transformer-Only model). This suggests that the Hybrid model can achieve virtually the same performance as the transformer-only model while being significantly more computationally efficient on small lightweight ViT models.

5.4 Comparison with Different Search and NAS Methods

Different Search Methods. We compare three types of search strategies: random search, evolution search Guo et al. (2020), and our coarse-to-fine search. Evolution search operates in two phases: (i) Crossover: Two randomly selected candidates are crossed to produce a new one. (ii) Mutation. A candidate is chosen randomly and mutated with a probability of 0.1 for each choice to produce a new candidate. Although no training is involved, the process is still time-consuming because the two phases are repeated to generate enough new candidates to obtain those that meet the given architecture constraints. As shown in Table 6, our coarse-to-fine search method can achieve better results (71.3 vs. 70.7 Top-1 accuracy) compared to the previous evolution search with much less training time.

Different NAS Methods Table 7 shows a comparison of our method to several ViT-based NAS methods, namely, ASViT Chen (2022), GLiT Chen et al. (2021a), and Autoformer Chen et al. (2021d). Our model offers an excellent balance of efficiency and performance. With only 6.3M parameters and 1.4G FLOPs, AutoViT_S achieves a Top-1 Accuracy of 79.2%. Although slightly lower than ASViT's accuracy (81.2%), AutoViT_S accomplishes this with much less computational demand, emphasizing the model's efficiency. Under similar FLOPs, the Top-1 accuracy of our AutoViT_S is noticeably higher than GLiT (76.3%) and Autoformer (74.7%).

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 ranges to handle the huge search space in ViT and mitigate conflicts in the weight-sharing of sub-networks. Further, 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 current experiments are limited to classification tasks. Future work includes extending our framework to other vision tasks such as object detection. This work is scientific in nature, we do not believe it has potential negative societal impacts.

References

Arash Amini, Arul Selvam Periyasamy, and Sven Behnke. T6d-direct: Transformers for multi-object 6d pose direct regression. *arXiv preprint arXiv:2109.10948*, 2021.

Anonymous. NASVit: Neural architecture search for efficient vision transformers with gradient conflict aware supernet training. In *Submitted to The Tenth International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=Qaw16njk6L`. under review.

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

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

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

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, pp. 213–229. Springer, 2020.

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

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

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

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

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

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

Wuyang et al. Chen. Auto-scaling vision transformers without training. In *ICLR*, 2022.

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

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

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

Linhui Dai, Hong Liu, Hao Tang, Zhiwei Wu, and Pinhao Song. Ao2-detr: Arbitrary-oriented object detection transformer. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.

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

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

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

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

Lei Ding, Dong Lin, Shaofu Lin, Jing Zhang, Xiaojie Cui, Yuebin Wang, Hao Tang, and Lorenzo Bruzzone. Looking outside the window: Wide-context transformer for the semantic segmentation of high-resolution remote sensing images. *arXiv preprint arXiv:2106.15754*, 2021.

Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, and Nenghai Yu. Peco: Perceptual codebook for bert pre-training of vision transformers. *arXiv preprint arXiv:2111.12710*, 2021.

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

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021.

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

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

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

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

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

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

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

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

Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. BossNAS: Exploring hybrid CNN-transformers with block-wisely self-supervised neural architecture search. In *ICCV*, 2021a.

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

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

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

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

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

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

Sachin Mehta, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. 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*, pp. 9190–9200, 2019.

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

Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

Manas Sahni, Shreya Varshini, Alind Khare, and Alexey Tumanov. Compofa: Compound once-for-all networks for faster multi-platform deployment. *arXiv preprint arXiv:2104.12642*, 2021.

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

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

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

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

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

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

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

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

Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. *arXiv preprint arXiv:2112.09133*, 2021.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

Dongkuan Xu, Subhabrata Mukherjee, Xiaodong Liu, Debadeepta Dey, Wenhui Wang, Xiang Zhang, Ahmed Hassan Awadallah, and Jianfeng Gao. Few-shot task-agnostic neural architecture search for distilling large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=GdMqXQx5fFR`.

Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5791–5800, 2020.

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

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

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

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

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

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6881–6890, 2021.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13001–13008, 2020.