FLORA: Fine-grained Low-Rank Architecture Search for Vision Transformer

Chi-Chih Chang[1,*], Yuan-Yao Sung[1,*], Shixing Yu[2,3,*], Ning-Chi Huang[1], Diana Marculescu[2], Kai-Chiang Wu[1]
National Yang Ming Chiao Tung University[1], University of Texas at Austin[2], Cornell University[3]
brian1009.en08@nycu.edu.tw, sungyuanyao@gmail.com,
sy774@cornell.edu, dianam@utexas.edu, {nchuang, kcw}@cs.nctu.edu.tw

Abstract

Vision Transformers (ViT) have recently demonstrated success across a myriad of computer vision tasks. However, their elevated computational demands pose significant challenges for real-world deployment. While low-rank approximation stands out as a renowned method to reduce computational loads, efficiently automating the target rank selection in ViT remains a challenge. Drawing from the notable similarity and alignment between the processes of rank selection and One-Shot NAS, we introduce FLORA, an end-to-end automatic framework based on NAS. To overcome the design challenge of supernet posed by vast search space, FLORA employs a low-rank aware candidate filtering strategy. This method adeptly identifies and eliminates underperforming candidates, effectively alleviating potential undertraining and interference among subnetworks. To further enhance the quality of low-rank supernets, we design a low-rank specific training paradigm. First, we propose weight inheritance to construct supernet and enable gradient sharing among low-rank modules. Secondly, we adopt low-rank aware sampling to strategically allocate training resources, taking into account inherited information from pre-trained models. Empirical results underscore FLORA's efficacy. With our method, a more fine-grained rank configuration can be generated automatically and yield up to 33% extra FLOPs reduction compared to a simple uniform configuration. More specific, FLORA-DeiT-B/FLORA-Swin-B can save up to 55%/42% FLOPs almost without performance degradtion. Importantly, FLORA boasts both versatility and orthogonality, offering an extra 21%-26% FLOPs reduction when integrated with leading compression techniques or compact hybrid structures. Our code is publicly available at https://github.com/shadowpa0327/FLORA.

1. Introduction

The transformer architecture [35] has dominated natural language processing (NLP) tasks with impressive results. Though intuitively, the transformer model seems inept to the special inductive bias of space correlation for image-oriented tasks, it has proved its capability on vision tasks with comparable results to convolutional neural network (CNN) [11]. Since their inception, vision transformers (ViT) and their variants have shown great potential for image classification [49], object detection [36], and semantic segmentation [25]. However, the ViT requires a large number of parameters and high computational cost to obtain higher accuracy, making it unsuitable for edge computing. That is mainly due to the stack of self-attention modules that suffer from quadratic complexity with regard to the input size, among other factors. Hence, research on efficient transformer models has become more important recently.

Earlier works on compressing ViTs mainly follow the techniques for compressing NLP models, ranging from unstructured pruning [50], attention head/structured pruning [8, 46], token pruning [20, 29, 41]; to knowledge distillation [18, 34] and quantization [26, 48].

Aside from the above-mentioned directions, another important category of method that employ efficiency in neural network (NN) structure is low-rank approximation. In the case of 2D low-rank approximation, singular-value decomposition (SVD) minimizes the Frobenius norm of the difference between the original matrix and the approximated matrix. Yet, SVD cannot be directly utilized for convolutions in CNNs because weights need to be represented by higher-dimensional (*e.g.,* 4D) tensors [22]. Special design [19, 21] is developed for CNNs by decomposing them into multiple consecutive tensors. However, there is still significant accuracy drop even after fine-tuning as training is performed on the transformed network structure with consecutive tensors without activation functions in-between. As a result, convergence can be degraded due to vanishing or exploding gradients [22]. This obstacle is largely alleviated in ViTs since 90% of total parameters and operations (see the supplemental materials for detailed analysis) are linear modules and conduct matrix multiplication. Linear modules can be decomposed into just two consecutive matrices. To put it in another way, ViTs are much more friendly than CNNs when incorporating low-rank decomposition in their

structure. Meanwhile, low-rank decomposition considers redundancy in deep neural networks as noise that contains a very small percentage of variance. Hence, it's different from general pruning methodologies for NN compression, in the way that the overall structural dimension and information flow will not be affected or truncated through low-rank guided compression techniques.

To effectively integrate low-rank approximations without compromising network performance, the central challenge lies in identifying a suitable set of rank settings. In the context of a Vision Transformer (ViT) with N linear modules and a maximum rank value of M, the potential rank settings explode to an immense number of M^N. Typically, M takes values in the order of a few hundred, leading to an exceedingly vast array of choices. However, existing methods that deal with low-rank approximation for transformers often resort to heuristic rank choices [27] or enforce uniform ranks across all compression targets [37], due to the absence of an automated rank selection strategy. Clearly, there is a strong need for an automatic rank selection approach to reduce the resource-intensive search process and mitigate the potential for suboptimal outcomes.

To surmount this challenge we present an innovative approach: reimagining the rank selection process as a Neural Architecture Search (NAS) problem. This new viewpoint is based on two main insights.

First, finding the best rank is a lot like choosing the right architecture for low-rank models. This similarity arises because picking the rank essentially how the corresponding low-rank architecture is built underneath.

Second, a property of SVD-based low-rank architectures, specifically the inheritance of top-r eigenvectors from pretrained. For different rank settings r_i and r_j, where $r_i < r_j$ the corresponding low-rank architecture share a significant portion of information from pretrained weights at the initial state. Such intrinsic property offers a great opportunity for us to train these candidates jointly like the way the supernet is optimized via weight sharing strategy in One-Shot NAS [14,30]. Building upon the well-recognized principles of One-Shot NAS, we introduce Fine-grained Low-Rank Architecture Search, dubbed as FLORA, to search for optimal low-rank architecture for ViT. Our main contributions are outlined as follows:

- We have discovered a captivating correlation between rank selection and One-Shot NAS and propose a first-of-its-kind low-rank architecture search regime for vision transformer.
- To realize our vision, we propose a low-rank aware filtering policy specifically crafted to eliminate candidates exhibiting subpar performance. By mitigating interference from less promising alternatives, this approach directly addresses the training challenges a supernet encounters due to the vast low-rank search space, thereby enhancing the efficiency and effectiveness of the overall NAS process.
- Building on this, we architect a unique supernet training paradigm tailored for low-rank structures, leveraging the inherent properties of SVD. This promotes both parameter and gradient sharing, significantly accelerating supernet convergence. As a result, subnetworks emerge with competitive performance, primed for direct deployment without exhaustive retraining.
- We conduct a series of experiments to demonstrate FLORA's orthogonality to other compression techniques and its generalization capability on ViT variants, which provides a new perspective on high ratio compression for ViT. Extensive experiments further demonstrate competitive results on ImageNet-1k when compressing DeiT and Swin-Transformer. For instance, FLORA-DeiT-B and FLORA-Swin-B achieve FLOPs reductions of 55% and 46%, respectively, on their corresponding backbones with negligible accuracy degradation.

2. Related Work

2.1. Transformer Compression

Compression methods for transformers can be broadly categorized into: pruning, token reduction and efficient architecture design. Pruning techniques are proposed to alleviate the high computational cost and memory usage by removing the redundant weights in the transformer-based models. VTP [50] reduced the number of embedding dimensions by extending the network slimming approach [24] to ViTs. [13, 15] proposed to skip the inessential layers to obtain a shallow model. Similarly, WDPruning [42] removed the less significant channels of the linear projection by using a neural-network-based saliency predictor. For the token-reduction-based techniques, [29, 32] hierarchically remove the redundant patches, thus reducing the computational overhead by slimming the input features.

Aside from above, some other works dedicated on design a efficient architecture directly by introducing CNNs to form a hybrid structure. MobileViT [28] mixed global processing in transformers with convolution, which learns better representations with fewer parameters and simple training recipes.

Recent work endeavored to combine multiple pruning strategies into a unified framework, *i.e.*, considering multiple dimensions simultaneously. For instance, UVC [46] pruned the heads in MHSA, channels in linear projection, and considered layer skipping. Meanwhile, MDC [16] jointly optimized an extra dimension, the number of patches.

2.2. Low-Rank Approximation

Aside from the model compression technique mentioned in the previous section, the low-rank matrix factorization on weights is also an effective methodology to reduce computational burden. In CNN-based models, [31] split the convolution kernel into two small ones with fixed rank. Some other works studied rank selection to generate finer-grained low-rank approximation. [40] selected the rank base on thresholding. [17] introduced rank-based cost function and formulated a constraint optimization problem to decide the rank selection during training.

Low-rank approximation (LRA) for transformer-based models can mainly be divided into two categories: (1) LRA for attention matrix (ScatterBrain [5], OmniNet [33]) and (2) LRA for linear embedding (LRT [37]). The former is mainly designed for transformers in NLP whose input sequences are relatively long. When the sequence length (patch number) is small (*e.g.*, ViT), the performance gain is relatively minor, as shown in the original paper of ScatterBrain. As for the latter one, LRT applies LRA with uniform-rank config and demonstrates its potential on Transformer for speech recognition tasks. Besides, we empirically found that the sensitivity to the performance loss concerning rank level is different among each linear embedding layer. Therefore, in this work, we set up a novel paradigm based on One-Shot NAS to explore the immense rank selection space of ViT more thoroughly and reduce the potential sub-optimality incurred by the simple manual setting.

2.3. Neural Architecture Search

One-Shot NAS [2, 4, 12, 14, 23, 30] has been proposed to efficiently discover architectures using weight-sharing strategies for CNN. DARTS [23] frames NAS as a constrained optimization problem and employs differentiable methods to optimize both network parameters and architecture parameters jointly. Interestingly, some prior efforts have adapted DARTS [39, 47] for low-rank architecture search within CNNs. However, such DARTS-based methods often requiring retraining for robust performance, which might be expensive as the number of deployment considerations increase [3].

To surmount the challenge, two-staged based NAS [3, 3, 14] decouple the training and searching. A supernet is first trained, followed by the application of an evolutionary algorithm to identify the optimal architecture.

Focusing on the vision transformer, AutoFormer [7] extends the two-staged based NAS paradigm [45], integrating a weight entanglement strategy to seek the optimal ViT architecture. GLiT [6] further introduces the locality module, incorporating CNN-correlated features into the search space, thus reducing computational costs and explicitly modeling local correlations between patches.

Unlike traditional NAS methods that primarily focus on parameters like kernel size, embedding dimensions, and head numbers in transformers and CNNs, our work centers on the unique low-rank architecture search space. This search space is vast, with each module potentially presenting hundreds of candidate rank settings. Due to the lack of a specifically designed solution for supernet construction, the search process can be burdened by computational overheads and undertraining issues. In this work, we aim to address these challenges. We introduce an optimized One-Shot NAS approach, specially crafted for low-rank architecture selection, to fully harness the potential of low-rank approximation in optimizing ViT.

3. Preliminary

One-Shot NAS In this work, we focus on the two-staged based One-Shot NAS paradigm. First, the architecture search space $\mathcal{A}$ is encoded into an over-parameterized supernet $\mathcal{N}(\mathcal{A}, W)$, where W stands for the weight of supernet and W is shared among candidate architectures (*i.e.*, low-rank architectures $a \in \mathcal{A}$). Then, we train the supernet by maximizing the objective function:

$$W_{\mathcal{A}} = \underset{W}{\operatorname{argmin}} \mathbb{E}_{a \sim \Gamma(\mathcal{A})}[\mathcal{L}(\mathcal{N}(a, W(a))], \tag{1}$$

where $W_{\mathcal{A}}$ is the weights of the supernet, $a \in \mathcal{A}$ is a rank configuration, $W(a)$ is weights of a, $\mathcal{N}(a, W(a))$ denotes the corresponding subnetwork, and $\mathcal{L}$ stands for the training loss.

Second, with the well-trained supernet, we can leverage it as a confident proxy to guide the searching algorithm such as evolutionary algorithm (EA) to search for optimal architecture that maximize the objective function (*e.g.*, accuracy) while satisfying the target constraint (*e.g.*, FLOPs). The objective function can be formulated as:

$$\begin{aligned} a^* &= \underset{a \in \mathcal{A}}{\operatorname{argmax}} \mathrm{ACC}_{\mathrm{val}}(\mathcal{N}(a, W_{\mathcal{A}}(a))) \\ \text{s.t.} \quad & \mathcal{F}_{lower} \leq \mathcal{F}(a) \leq \mathcal{F}_{upper}, \end{aligned} \tag{2}$$

where $\mathrm{ACC}_{\mathrm{val}}$ is the accuracy on validation set, $\mathcal{F}(a)$ denotes the FLOPs of the subnetwork with the configuration a, $\mathcal{F}_{lower}$ and $\mathcal{F}_{upper}$ are the lower bound and upper bound of FLOPs constraints, respectively.

4. Methodology

4.1. Mappging Rank Selection into NAS

Our primary goal is to maximize the benefits of low-rank approximation within transformers. This entails identifying the optimal rank setting r and then generating the associated low-rank modules for each linear component. In pursuing this, we observed a clear correlation between rank selection and One-Shot NAS, leading to two key insights:

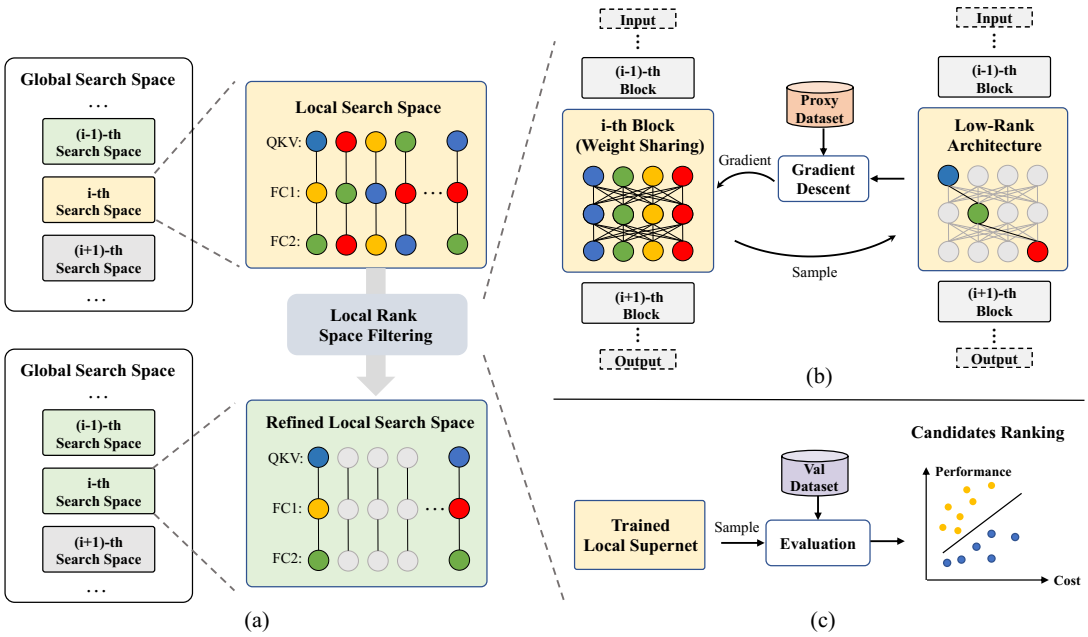

Figure 1. Illustration of low-rank aware candidate filtering. **(a)**: Filtering is applied sequentially to each block. **(b)**: For the i-th transformer block, we construct a local supernet by substituting the i-th block with a weight-sharing superblock that encompasses the local low-rank architecture. **(c)**: Leveraging the trained supernet as a performance estimator, we filter the local rank space, considering the trade-off between performance benefits and computational costs.

- In low-rank approximation, given a rank setting r, we decompose the linear modules into two matrices: $U_r \in \mathbb{R}^{m\times r}$ and $V_r \in \mathbb{R}^{n\times r}$. Under this scenario, deciding on an optimal rank choice is equivalent to choosing architectural components in network design.
- The rank-r low-rank approximation inherits the top-most r significant eigenvectors from full-rank pre-trained weight matrices. When considering two rank settings r_i and r_j, where $r_i < r_j$, it is evident that the set of eigenvectors for r_i is a proper subset of those of r_j. During finetuning, this subset property suggests that the low-rank matrices U_{r_i} and V_{r_i} of rank setting r_i can be approximately inherited from U_{r_j} and V_{r_j}, respectively. This is because their initial states are identical, with the top r_i eigenvectors holding a significant portion of information from the pretrained weight matrices. This characteristic provides a perfect link for these candidates to be trained jointly, which aligns well with One-Shot NAS and its weight-sharing concept.

Based on these insights, we propose to map the rank selection problem into NAS and introduce the Fine-grained Low-Rank Architecture Search, abbreviated as FLORA, to effectively identify fine-grained low-rank architecture.

Challenge While the mapping into One-Shot NAS presents a promising pathway, there are still challenges we must overcome to actualize a sound and effective solution for rank selection. Traditional One-Shot NAS methodologies often focus on a discrete search space with around 3-5 choices per block, which is deemed manageable in terms of computational demand. However, when we consider the realm of rank selection for low-rank approximation, the scenario becomes vastly more intricate, since the candidate rank choices is usually larger than hundreds.

Earlier studies focused on low-rank approximation for CNNs suggest a simplification of this astounding search space. They advocate for ranks adhering strictly to a constant multiple, often 32 or 64, aiming to strike an optimal balance between search cost and granularity [39]. However, when these methods are applied to architectures like ViT, the sheer scale of the search space remains overwhelming. For instance, the classic DeiT-B, with its embedding dimension of 768 and a constant multiple of 64, still presents up to 12 rank choices for each linear module.

This expansive choice set introduces two primary challenges. Firstly, without a well-designed strategy to construct and train the supernet, subnetworks may risk undertraining or face weight-coupling issues [1, 9]. This can lead to biased estimations of network performance. Secondly, once an approximate model has been identified, retraining is often essential to refine its performance – an additional computational overhead that is not desirable, especially in compression scenarios.

To overcome the design challenges mentioned above, we

offer a series of techniques to enhance the performance of low-rank supernets, including the filtering strategy to identify weak candidates to prevent from possible interferencing (Sec. 4.2) and alongside low-rank aware paradigm for supernet consruction and training crafted for resource efficiency and accelerated convergence based on the observation of inherent property of low-rank approximation (Sec. 4.3).

4.2. Low-Rank Aware Candidate Filtering

Supernet performance is deeply linked to the quality of its search space. When employing low-rank approximation, aggressive rank settings can sometimes lead to irreversible information loss. Conversely, architectures with conservative rank settings might not deliver any substantial benefits in terms of computational efficiency. The weight-sharing nature of supernets further complicates this, as tightly coupled subnetwork weights can negatively influence promising architectures, leading to interference and compromised performance.

A seemingly straightforward solution would be to discard these unsuitable configurations. However, the vastness of the low-rank search space renders such a direct approach both cumbersome and computationally taxing. To address this, we draw on a key observation: architectures that underperform at the local level (withing one transformer block) often falter in a global context (entire low-rank architecture). With this in mind, we introduce a bi-level filtering mechanism, called *low-rank aware candidate filtering*, to identify architectures that strike a balance between accuracy and computational demand. The flow of our algorithm is illustrated in Fig. 1.

Local Level Filtering Our strategy commences at the local level, focusing on individual transformer blocks. Technically, we create a local supernet that encompasses the low-rank architecture of the current transformer block, leaving other blocks uncompressed. This local supernet is then trained on a proxy dataset—a subset of the original training data.

To steer our selection process, we employ the Precision Cost Ratio, defined as:

$$\mathcal{M}(a) = \lambda * \mathcal{P}(a) - \mathcal{F}(a) \tag{3}$$

Here, $\mathcal{P}(a)$ and $\mathcal{F}(a)$ respectively denote the classification accuracy and computational overhead associated with a low-rank architecture a. The parameter λ adjusts the balance between precision and cost. Utilizing this metric, we rank the low-rank architecture on a block level and retain only the top-k candidates.

Global Level Integration According to the results of local filtering, we then generated the potential global architectures based on the cartesian product. In essence, any global architecture that houses a locally discarded component is eliminated. By doing so, we effectively weed out underperforming candidates, giving rise to a optimized search space, which is smaller and more friendly for supernet construction and training.

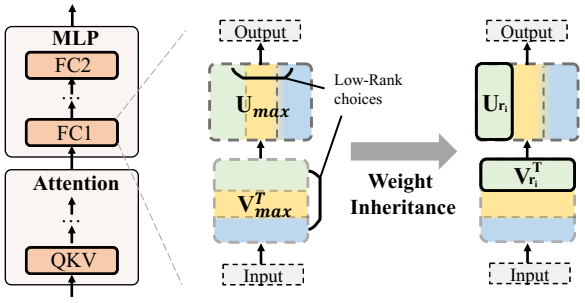

Figure 2. Illustration of weight inheritance technique for supernet construction. All rank architecture of a specific linear modules are encompassed within a superblock. For a designated rank settings, the corresponding rank architecture will be generated by inheriting from superblock

4.3. Boosting Low-Rank Supernet

Achieving a superior supernet performance involves multiple aspects. While refining the search space is crucial, the way subnetworks are incorporated and trained within the supernet is equally important. The successful convergence of the supernet, and its ability to prepare low-rank subnets for immediate use without additional training, depends on both the integration method and the training strategy. To address these challenges, we introduce the *Low-Rank Aware Training Paradigm* (LRAT). Tailored for low-rank architectures, LRAT combines two main components: *Weight Inheritance*, which ensures a suitable integration of various rank choices, and *Low-Rank Aware Sampling*, a training approach that adjusts to the needs of different rank architectures.

Weight Inheritance For a linear component with N candidate rank choices $\mathbb{C} = \{r_1, r_2, ..., r_N\}$ and we can yield corresponding low-rank module set $\mathbb{S} = \{(U_{r_1}, V_{r_1}), (U_{r_2}, V_{r_2}), ..., (U_{r_N}, V_{r_N})\}$. Our goal is to design a suitable technique to integrate all of these candidates into a choice block so that our supernet can be constructed.

In our earlier discussion (Sec. 4.1), we explored how top eigenvectors overlap across different rank choices in low-rank approximations. Specifically, for a rank r_i, the top eigenvectors, originating from the full-rank pretrained weight matrices, capture a significant chunk of information from the initial state. Furthermore, these eigenvectors for r_i act as a subset to the eigenvectors of a higher rank r_j where $r_i < r_j$. This suggests that as we navigate through different rank settings, there is both reusability and overlap

of information from the initial state across different ranks. Consequently, instead of handling each low-rank module in isolation, we could co-train the corresponding modules by jointly sharing their parameters and gradients to boost the convergence.

Building on these insights and drawing inspiration from the shared convolutional kernels in CNN NAS practices [30, 45], we present the *weight inheritance* method. As illustrated in Fig. 2, every low-rank module is encapsulated within the largest one, expressed as:

$$U_{r_i} = U_{r_N}[:, : r_i], \quad V_{r_i} = V_{r_N}[:, : r_i] \tag{4}$$

For a specific rank r_i, the values of U_{r_i} and V_{r_i} are inherited from the top-r_i columns of U_{r_N} and V_{r_N} respectively. During the backward pass, the gradients of U_{r_i} and V_{r_i} are updated back to the corresponding sub-matrices in U_{r_N} and V_{r_N}. Within each choice block, the candidate with the largest rank is set to the size of the uncompressed model's factorized weights and with SVD.

It's evident that each sub-structure is a subset of the structures with higher rank levels, culminating in the supermatrix, as shown below:

$$\begin{gathered} U_{r_i} \subseteq U_{r_k}, \quad V_{r_i} \subseteq V_{r_k} \\ \forall k \in \{i+1, i+2, ..., N\} \quad \forall r_i \in \mathbb{C} \end{gathered} \tag{5}$$

With shared weights among low-rank modules, gradient information can be efficiently utilized across groups. This mutual gradient update strategy ensures that all candidate low-rank architectures are trained concurrently, bolstering convergence during the supernet training phase.

Low-Rank Aware Sampling Contrary to prior works [7, 14] that train the supernet from scratch, our approach initializes the supernet with weights from a pretrained uncompressed model. In this context, subnetworks are seen as low-rank approximated networks for different rank choices. This introduces a new perspective on sampling during training: it's suboptimal to sample subnetworks using a uniform distribution. Instead, our design for a new sampling distribution is guided by two key observations:

- Architectural search algorithms are generally complex and hard to train. For a pre-selected search space, it's computationally arduous to thoroughly train every substructure. For the entire system to achieve peak performance and consistent convergence, training resources must be judiciously allocated to different subnetworks, warranting a non-uniform distribution.

- Low-rank architectures, especially those of lower ranks, derive varying degrees of information from pretrained weights. Such inheritance can be perceived as the architecture having undergone preliminary training. As a result, during supernet training, it's crucial to divert more resources (like sampling probability) towards these less-informed structures, ensuring balance.

With the above insights in mind, we introduce a non-uniform path sampling strategy that favors smaller rank choices within each low-rank linear layer. Formally, let's define the rank choice for a low-rank linear layer by a random variable X. Its Probability Mass Function (PMF) of X is formulated as:

$$p_X(r) = P(X = r) = \frac{\frac{1}{r}}{\sum_{r' \in \mathbb{C}} \frac{1}{r'}}, \quad r \in \mathbb{C} \tag{6}$$

Recall that $\mathbb{C}$ denotes the rank choice set of a specific low-rank linear layer. The prior distribution of selecting a sequence of rank choice would be $\Gamma(\mathcal{A}) = P(X_1 = r_{k_1}, ..., X_i = r_{k_i}, ..., X_l = r_{k_l})$, where X_i, r_{k_i} denotes the random variable of i-th choice block and its correlated rank choice, respectively.

5. Experiments

We trained and tested FLORA on the ImageNet-1k [10] dataset using several representative ViT models, such as DeiT [34] and Swin Transformer [25], as our pre-trained backbones. In line with earlier NAS practices [43, 44], our supernet is trained under the in-place distillation strategy (e.g., under the supervision of the uncompressed model itself). We employed the evolutionary algorithm to search for the target architecture. For detailed information on hyperparameters used in training and supernet configuration, please refer to the supplemental materials.

5.1. General Comparison

We compare our results with state-of-the-art ViT pruning methods, ranging from input sequence reduction (DynamicViT [29]), weight pruning (WDPruning [42], S^2ViTE [8]), and multi-dimension pruning (UVC [46]). It is worth mentioning that DynamicViT and UVC both include the knowledge distillation in their methodologies, which is identical to our method.

The results are shown in Tab. 1. Compared to channel pruning methodologies (WDPruning [42]), we achieve around 1% accuracy improvement under the same or smaller level of computation reduction and model size on every backbone, demonstrating the effectiveness of searching for a low-rank subnetwork against structure pruning techniques. Besides, when considering the sequence reduction-based methods (DynamicViT [29]), FLORA achieves comparable performance under with better FLOPs savings with different optimized targets. For DeiT-B/Swin-B, we can either enjoy 20%/21% additional FLOPs reduction with competitable or even higher top-1 accuracy. Besides, FLORA further demonstrates its superiority with not only FLOPs saving but an extra 60%/30%

Table 1. Comparison of FLORA with different ViT compression methods on ImageNet-1k dataset.

Method	FLOPs	FLOPs saving	Params	Params saving	Top-1 Acc
DeiT-S					
Baseline	4.7G	-	22.1M	-	79.8%
DynamicViT	3.4G	28%	23.1M	-	79.6%
SPViT	3.3G	30%	16.4M	26%	78.3%
WDPruning	3.1G	32%	15.0M	32%	78.6%
S^2ViTE	3.1G	32%	14.6M	34%	79.2%
UVC	2.7G	42%	-	-	79.4%
Ours	2.7G	42%	12.6M	43%	79.6%
DeiT-B					
Baseline	17.6G	-	86.4M	-	81.8%
S^2ViTE	11.8G	33%	56.8M	35%	82.2%
SPViT	11.7G	33%	62.3M	28%	81.6%
DynamicViT	11.2G	36%	87.2M	-	81.3%
WDPruning	11.0G	37%	60.6M	30%	81.1%
UVC	8.0G	55%	-	-	80.6%
Ours	8.0G	55%	37.9M	56%	81.8%
Swin-S					
Baseline	8.7G	-	49.6M	-	83.2%
DynamicViT	6.9G	20%	50.8M	-	83.2%
WDPruning	6.8G	22%	37.4M	26%	82.4%
SPViT	6.1G	30%	39.2M	30%	82.4%
Ours	6.1G	30%	34.8M	30%	82.8%
Swin-B					
Baseline	15.4G	-	87.8M	-	83.5%
DynamicViT	12.1G	21%	88.8M	-	83.4%
SPViT	11.4G	26%	68.0M	24%	83.2%
Ours	9.0G	42%	52.3M	41%	83.2%

model size reduction when compared to DynamicViT. Lastly, when compared to multi-dimensional compression methods, we achieve superior results on DeiT-Base, registering a 1.2% improvement in performance under the same FLOPs level. Overall, we can observe that FLORA shows competitive performance with SOTA methods on small models like DeiT-S and Swin-S while outperforming all the methods under comparison on large models like DeiT-B and Swin-B.

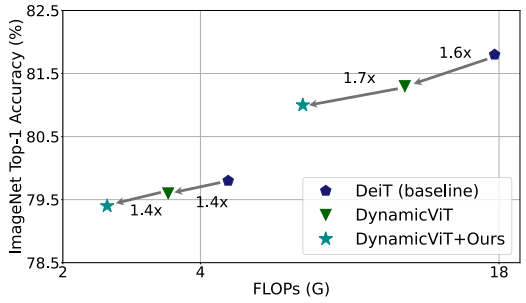

Figure 3. Results of integrating FLORA with DynamicViT.

Table 2. Experiments on TinyViT and integration with DynamicViT. DynamicViT is denoted as DyViT. Results show the orthogonality of FLORA on top of other compression methods

Model	Methods	GFLOPs	Top-1 Acc
TinyViT-21M	baseline	4.3G	83.1%
	Ours	**3.2G (-25%)**	**83.1% (-0%)**
DeiT-S	baseline	4.7G	79.8%
	DyViT	3.4G (-28%)	79.6% (-0.3%)
	DyViT + Ours	**2.3G (-49%)**	**79.3% (-0.6%)**
DeiT-B	baseline	17.6G	81.8%
	DyViT	11.2G (-36%)	81.3% (-0.5%)
	DyViT + Ours	**6.70G (-62%)**	**81.0% (-0.8%)**

5.2. Generality and Orthogonality

Results on the Compact Hybrid Transformer In the previous section, we have demonstrated the generality of FLORA on Transformer-only variants DeiT and Swin Transformer. Here we further show that our approach is agnostic to model architectures even on a hybrid structure (*i.e.*, CNN + Transformer). The results of FLORA a TinyViT-22M [38] with 224 x 224 input is shown in Tab. 2. The computational cost can be reduced by 25% without sacrificing performance on TinyViT, affirming the generality of our proposed FLORA.

Integrating with Other Compression Methods To show the orthogonality of our approach, we integrate the proposed FLORA with token reduction based compression method DynamicViT to achieve further compression ratio. Here, we use the uncompressed model itself for supervising. From Tab. 2, FLORA achieves 21%/26% more FLOPs reduction while only sacrificing 0.3% more performance degradation on DeiT-S/B. Fig. 3 provides an clearer view for the demonstration of orthogonality. On DeiT-B, when integrating FLORA with DynamicViT, the reduction on FLOPS is further improved by 1.7x over standalone DynamicViT. Moreover, on the small-scale model DeiT-S, we can also earn 1.4x more FLOPs improvement and a compelling 50% FLOPs savings in total. Thus, FLORA provides a new perspective that is orthogonal to the previous compression methods. By applying multiple compression techniques, ViT can be compressed to an appealing ratio with less than 50% of the original FLOPs and less than 1% accuracy drop.

5.3. Benefit of Non-Uniform Rank Setting

In this section, we study the impact of employing fine-grained, non-uniform rank configurations on the model's performance. We utilize DeiT-B as the backbone and proceed with training a low-rank supernet. To ensure fair comparison, all considered low-rank architectures are derived from this trained supernet. As illustrated in Fig. 4, the advantages of a non-uniform rank configuration become apparent. When comparing under the same accuracy level, a non-uniform rank configuration can achieve up to 33% additional FLOPs reduction relative to its uniform counter-

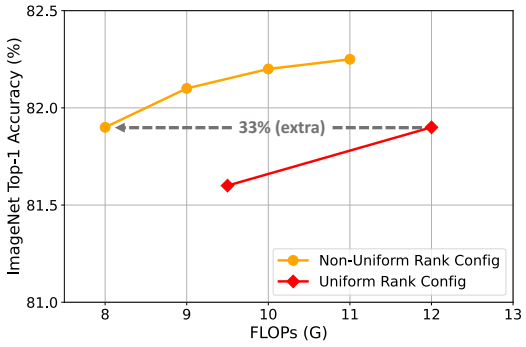

Figure 4. Performance difference of using different rank config for Low-rank approximation on DeiT-B.

part [37]. These findings underscore the value of layer-wise non-uniform rank configurations in the low-rank approximation for ViT and further emphasize to the proficiency of FLORA in uncovering these configurations.

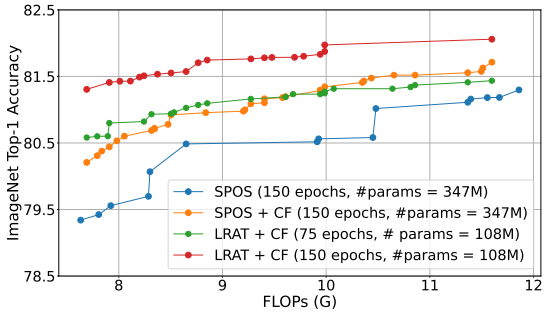

Figure 5. Pareto frontier of subnetworks sampled from low-rank supernet of DeiT-B with different strategies. #params denote the number of trainable parameters of the supernet. CF: Low-Rank Aware Candidate Filtering, LRAT: Low Rank Aware Supernet Training Paradigm

5.4. Ablation Study

In this section, we conduct an ablation study to analyze the impact of our *Low-Rank Aware Candidate Filtering* and the proposed *Low-Rank Aware Training Paradigm* on enhancing supernet quality. The results are visualized using Pareto frontier analysis in Fig. 5. Our experiments are based on the DeiT-B backbone. As a baseline, we employ the well-established One-Shot NAS method, SPOS [14], which was originally designed for CNN search spaces. We have adapted SPOS to accommodate the low-rank search space, while preserving its original settings. In the course of supernet construction, each low-rank module within the choice blocks is treated as an independent entity. We train the SPOS supernet under the same hyperparameters as ours. During the supernet's training phase, architectures are selected through uniform sampling.

Efficacy of Low-Rank Aware Candidate Filtering As elaborated in Section Sec. 4.2, our low-Rank aware candidate filtering focuses on pinpointing weaker architectural components and then uses this insight to generate the refined search space. The marked benefits of our filtering technique on supernet performance are illustrated in Fig. 5. Using the same training approach as the SPOS baseline, the performance of subnetworks across diverse FLOP levels sees a notable boost with the incorporation of our filtering approach. Notably, for subnetworks operating around the 8G FLOP mark, we observe a nearly 1% enhancement in Top-1 accuracy. This considerable enhancement underscores the proficiency of our low-rank aware candidate filtering in pinpointing and eliminating subpar architectures, effectively reducing the training interference from such suboptimal configurations.

Efficacy of Low-Rank Aware Training Paradigm Our approach to low-rank aware supernet training combines the *Weight Inheritance* technique with *Low-Rank Aware Sampling*, both specifically designed for low-rank scenarios. Notably, our supernet, trained for only 75 epochs using this approach, performs as well as the SPOS paradigm which requires 150 epochs, emphasizing a faster convergence. When trained for the full duration, our results sit on the Pareto frontier, demonstrating the efficacy of our paradigm.

6. Conclusion

In this work, we focus on the rank selection problem, exploring the integration of low-rank approximation into the Vision Transformer (ViT) architecture. We observe a notable similarity and alignment between the processes of rank selection and One-Shot NAS. Motivated by this observation, we introduce FLORA—an end-to-end framework based on NAS—to autonomously search for the fine-grained low-rank configurations. To bolster the effectiveness of this approach, we also introduce a series of specialized techniques tailored to enhance the quality of the low-rank supernet. Extensive experiments from our research demonstrates the potential of low-rank approximation as a effective compression technique for the optimization of transformers. For instance, on DeiT-B, with fine-grained rank configuration up to 55% FLOPs can be saved without performance drop. This discovery holds immense implications, particularly considering the current void in methods that automate rank selection for ViT.

Acknowledgements

This work was supported in part by NSTC Grant No. NSTC 112-2218-E-A49-019 (Taiwan), NSF CCF Grant No. 2107085 (US), ONR Minerva program, and iMAGiNE — the Intelligent Machine Engineering Consortium at UT Austin. We thank to National Center for High-performance Computing (NCHC) of National Applied Research Laboratories (NARLabs) in Taiwan for providing computational and storage resources.

References

[1] George Adam and Jonathan Lorraine. Understanding neural architecture search techniques. *CoRR*, abs/1904.00438, 2019. 4

[2] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. SMASH: one-shot model architecture search through hypernetworks. In *ICLR*, 2018. 3

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020. 3

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 3

[5] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. In *NeurIPS*, 2021. 3

[6] 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 *ICCV*, pages 12–21, 2021. 3

[7] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *ICCV*, pages 12250–12260, 2021. 3, 6

[8] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. In *NeurIPS*, pages 19974–19988, 2021. 1, 6

[9] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *ICCV*, pages 12219–12228. IEEE, 2021. 4

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 6

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

[12] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 2019. 3

[13] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *ICLR*, 2020. 2

[14] 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 *ECCV*, pages 544–560, 2020. 2, 3, 6, 8

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

[16] Zejiang Hou and Sun-Yuan Kung. Multi-dimensional vision transformer compression via dependency guided gaussian process search. In *CVPR*, pages 3668–3677, 2022. 2

[17] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán. Low-rank compression of neural nets: Learning the rank of each layer. In *CVPR*, 2020. 3

[18] Ding Jia, Kai Han, Yunhe Wang, Yehui Tang, Jianyuan Guo, Chao Zhang, and Dacheng Tao. Efficient vision transformers via fine-grained manifold distillation. *arXiv preprint arXiv:2107.01378*, 2021. 1

[19] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016. 1

[20] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Bin Ren, Minghai Qin, Hao Tang, and Yanzhi Wang. Spvit: Enabling faster vision transformers via soft token pruning. *CoRR*, abs/2112.13890, 2021. 1

[21] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V. Oseledets, and Victor S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015. 1

[22] Dongsoo Lee, Se Jung Kwon, Byeongwook Kim, and Gu-Yeon Wei. Learning low-rank approximation for cnns. *CoRR*, 2019. 1

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ICLR*, 2019. 3

[24] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, pages 2755–2763, 2017. 2

[25] 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 *ICCV*, pages 9992–10002, 2021. 1, 6

[26] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34:28092–28103, 2021. 1

[27] Xiuqing Lv, Peng Zhang, Sunzhu Li, Guobing Gan, and Yueheng Sun. Lightformer: Light-weight transformer using svd-based weight transfer and parameter sharing. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *ACL*, pages 10323–10335, 2023. 2

[28] Sachin Mehta and Mohammad Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *ICLR*, 2022. 2

[29] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*, pages 13937–13949, 2021. 1, 2, 6

[30] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path NAS: designing hardware-efficient convnets in less than 4 hours. In Ulf Brefeld, Élisa Fromont, Andreas Hotho, Arno J. Knobbe, Marloes H. Maathuis, and Céline Robardet, editors, *ECML PKDD*, 2019. 2, 3, 6

[31] Cheng Tai, Tong Xiao, Xiaogang Wang, and Weinan E. Convolutional neural networks with low-rank regularization. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016. 3

[32] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. *CoRR*, abs/2106.02852, 2021. 2

[33] Yi Tay, Mostafa Dehghani, Vamsi Aribandi, Jai Prakash Gupta, Philip Pham, Zhen Qin, Dara Bahri, Da-Cheng Juan, and Donald Metzler. Omninet: Omnidirectional representations from transformers. In *ICML*, 2021. 3

[34] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021. 1, 6

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017. 1

[36] Yixuan Wei, Han Hu, Zhenda Xie, Zheng Zhang, Yue Cao, Jianmin Bao, Dong Chen, and Baining Guo. Contrastive learning rivals masked image modeling in fine-tuning via feature distillation. *CoRR*, abs/2205.14141, 2022. 1

[37] Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, and Pascale Fung. Lightweight and efficient end-to-end speech recognition using low-rank transformer. In *ICASSP*, pages 6144–6148, 2020. 2, 3, 8

[38] Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. *CoRR*, abs/2207.10666, 2022. 7

[39] Jinqi Xiao, Chengming Zhang, Yu Gong, Miao Yin, Yang Sui, Lizhi Xiang, Dingwen Tao, and Bo Yuan. HALOC: hardware-aware automatic low-rank compression for compact neural networks. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *AAAI*, pages 10464–10472. AAAI Press, 2023. 3, 4

[40] Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. TRP: trained rank pruning for efficient deep neural networks. In Christian Bessiere, editor, *IJCAI*, 2020. 3

[41] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 1

[42] Fang Yu, Kun Huang, Meng Wang, Yuan Cheng, Wei Chu, and Li Cui. Width & depth pruning for vision transformers. In *AAAI*, pages 3143–3151, 2022. 2, 6

[43] Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *ICCV*, pages 1803–1811. 6

[44] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas S. Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, pages 702–717, 2020. 6

[45] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas S. Huang. Slimmable neural networks. In *ICLR*, 2019. 3, 6

[46] Shixing Yu, Tianlong Chen, Jiayi Shen, Huan Yuan, Jianchao Tan, Sen Yang, Ji Liu, and Zhangyang Wang. Unified visual transformer compression. In *ICLR*, 2022. 1, 2, 6

[47] Zhewen Yu and Christos-Savvas Bouganis. SVD-NAS: coupling low-rank approximation and neural architecture search. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2023, Waikoloa, HI, USA, January 2-7, 2023*, pages 1503–1512. IEEE, 2023. 3

[48] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization framework for vision transformers. *arXiv preprint arXiv:2111.12293*, 2021. 1

[49] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *CoRR*, 2021. 1

[50] Mingjian Zhu, Kai Han, Yehui Tang, and Yunhe Wang. Visual transformer pruning. *arXiv preprint arXiv:2104.08500*, 2021. 1, 2