Vision Transformer Compression and Architecture Exploration with Efficient Embedding Space Search

Daeho Kim[1] [0000−0003−2921−3168] and Jaeil Kim[2,⋆] [0000−0002−9799−1773]

[1] Department of Artificial Intelligence, Kyungpook National University
[2] School of Computer Science and Engineering, Kyungpook National University
{kdaeho27,threeyears}@gmail.com

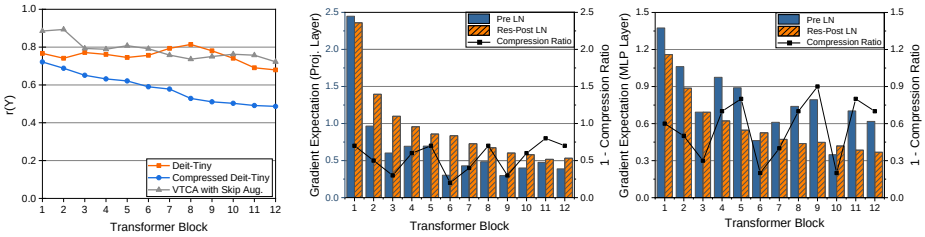
Abstract. This paper addresses theoretical and practical problems in the compression of vision transformers for resource-constrained environments. We found that *deep feature collapse* and *gradient collapse* can occur during the search process for the vision transformer compression. *Deep feature collapse* diminishes feature diversity rapidly as the layer depth deepens, and *gradient collapse* causes gradient explosion in training. Against these issues, we propose a novel framework, called VTCA, for accomplishing vision transformer compression and architecture exploration jointly with embedding space search using Bayesian optimization. In this framework, we formulate block-wise removal, shrinkage, cross-block skip augmentation to prevent *deep feature collapse*, and Res-Post layer normalization to prevent *gradient collapse* under a knowledge distillation loss. In the search phase, we adopt a training speed estimation for a large-scale dataset and propose a novel elastic reward function that can represent a generalized manifold of rewards. Experiments were conducted with DeiT-Tiny/Small/Base backbones on the ImageNet, and our approach achieved competitive accuracy to recent patch reduction and pruning methods. The code is available at *https://github.com/kdaeho27/VTCA*.

1 Introduction

Vision transformers have recently shown superior performance in learning long-range dependency property of sequential data and have attracted attention in various computer vision tasks. Modern architectures based on the transformer concept, such as ViT [10] and DeiT [29], are capable of learning significant visual representations from images and outperform traditional convolutional neural networks (CNNs) [18,16,13].

Despite the availability of vision transformers, such architectures have been demonstrated to be even more resource-intensive than CNNs and have deployment limitations in a resource-limited environment [36]. Due to the significant

⋆ Corresponding author



(a) Deep feature collapse (b) Gradient collapse

Fig. 1: (a) Deep feature collapse and (b) gradient collapse occurred during search process for compression.

structural differences between CNNs and ViT, using successful CNN compression methods [19,15,14] for ViT is problematic. Previous studies for the transformer compression include pruning [7,36,37], neural architecture search (NAS) [6], and patch reduction [26,23,24]. Most studies compress the number of heads, the hidden dimensions of the multi-layer perceptron (MLP) layers, and dropping blocks.

However, we found that *deep feature collapse* and *gradient collapse* occur during the compression process, as shown in Figure 1. *Deep feature collapse* denotes rapidly diminishing feature diversity as the transformer block deepens. This phenomenon occurs as the number of heads is compressed. As shown in Figure 1a, the feature diversity $r(Y)$ decreases rapidly with compression of the number of heads of DeiT-Tiny. *Gradient collapse* refers to changes in the scale of the gradient as the compression rate changes. Figure 1b shows the gradient expectation of hidden dimensions in MLP and QKV dimensions in self-attention modules. With a Pre-LN transformer, the gradient scale changes according to the compression rate, which causes gradient exploding during training.

To prevent the collapse problems, this paper aims to establish compression and architectural search jointly. We call this Vision Transformer Compression and Architecture exploration (VTCA). Figure 2 illustrates the overall structure of VTCA: it compresses hidden dimensions in the MLP as well as the number of heads in the self-attention module, and searches the architecture to add cross-block skip augmentation and layer normalization under knowledge distillation. To search for the optimized architecture, we propose a novel search process based on Bayesian optimization (BO) for vision transformer compression, as shown in Figure 3.

Our main contributions are as follows:

1. We formulate *deep feature collapse* and *gradient collapse* as problems occurring during the compression process for the vision transformer. To alleviate these problems, we propose a new framework based on BO, called VTCA, that integrates compression and architecture search.

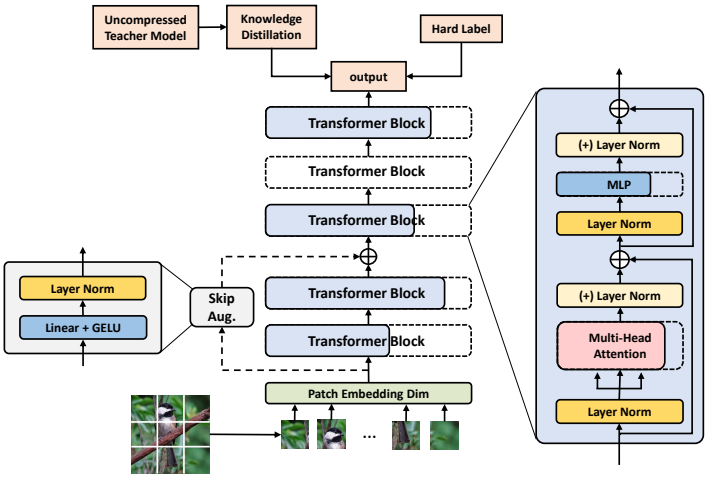


Fig. 2: The overall structure of VTCA, integrating compression and architecture exploration strategies: (1) **block-wise removal and shrinkage** —we compress self-attention head numbers, hidden dimension of MLP module, and blocks; (2) **cross-block skip augmentation**; (3) **addition of Res-Post layer normalization**; under knowledge distillation. For more details on the search space, see Section 3.3

.

2. For efficient space search, we propose a normalized cross-entropy score (NCE score) with training speed estimation to define as a reward.
3. We propose an elastic reward function including compression rate and NCE score to evaluate compressed architectures. The reward function is represented as generalizing both naive and N2N (Network to Network) [1] reward functions, and it can control a trade-off between compression rate and accuracy.
4. Experiments are conducted with popular variants of ViT on ImageNet; our method performs better than or comparably with existing methods.

2 Preliminaries and Motivation

2.1 Vision Transformer

Follwing the success of transformer architectures in natural language processing (NLP) tasks [30,8], recent approaches such as ViT [10] and DeiT [29], have been introduced for computer vision tasks. The ViT block consists of a multi-head self-attention (MSA) and MLP modules with layer normalization [3] placed in front of each module. An input image is split into N patches and each patch is projected into a d-dimensional vector. Given the feature $Y_l \in \mathbb{R}^{N \times d}$ in l-th layer,

the MSA module can be defined as:

$$\mathrm{MSA}(Y_l) = \mathrm{Concat}([A_{lh} Y_l W^v_{lh}]^{H_l}_{h=1}) W^o_l \tag{1}$$

$$A_{lh} = \mathrm{Softmax}\Big(\frac{(Y_l W^q_{lh})(Y_l W^k_{lh})^T}{\sqrt{d_h}}\Big) \tag{2}$$

where $A_{lh} \in \mathbb{R}^{N \times N}$ is self-attention map, $W^v_{lh} \in \mathbb{R}^{d \times (d/H)}$ is projection matrix in the h-th head, $W^o_l \in \mathbb{R}^{d \times d}$ is the output projection matrix. $W^q_{lh} \in \mathbb{R}^{d \times (d/H)}$ and $W^k_{lh} \in \mathbb{R}^{d \times (d/H)}$ are the query and value projection matrices, respectively.

The MLP module consists of two linear projections and extracts features from each patch independently. Given the MLP input feature $Z_l \in \mathbb{R}^{N \times d}$, the MLP can be defined:

$$\mathrm{MLP}(Z_l) = \sigma(Z_l W^{1,l} + b^{1,l}) W^{2,l} + b^{2,l} \tag{3}$$

where $W^{1,l} \in \mathbb{R}^{d \times d_m}$ and $W^{2,l} \in \mathbb{R}^{d_m \times d}$ are weights in the MLP module and σ is the non-linear activation function. The MLP and MSA modules are alternately stacked to construct a vision transformer model.

Most studies have focused on compressing or pruning the number of heads and the hidden dimension of MLP [6,34,36] at each block. We found that *deep feature collapse*, in which feature diversity diminish rapidly as the transformer block deepens, and *gradient collapse*, in which gradient expectation changes depending on the compression rate, occurred during compression.

2.2 Deep Feature Collapse

Feature collapse is defined as the occurrence of hard to distinguish features among patches in a layer as the block depth increases [27,9]. To distinguish from the *feature collapse* that occurs in vision transformers, we define *deep feature collapse* as occurring while compressing the number of heads H_l as the transformer block deepens.

Given an output feature Y_l in the l-th layer, feature diversity is measured as the difference between the features and the rank-1 matrix [9]:

$$r(Y_l) = \|Y_l - \mathbf{1}\mathbf{y}_l^T\|, \text{ where } \mathbf{y}_l^T = \mathrm{argmin}_{\mathbf{y}_l'} \|Y_l - \mathbf{1}\mathbf{y}_l^T\| \tag{4}$$

where $\|\cdot\|$ is an ℓ_1, ℓ_∞-composite norm. The feature diversity $r(Y_l)$ of the architecture in which the number of heads is compressed decreases rapidly as the block deepens, as defined by the following theorem established in [9].

Theorem 1. *Given a transformer model in which the MSA and MLP modules are stacked, the feature diversity $r(Y_l)$ in the l-th layer is bound by that of input Y_0, i.e.,*

$$r(Y_l) \downarrow \leq \Big(\frac{4H_l \downarrow \gamma\lambda'}{\sqrt{d}}\Big)^{\frac{3^l-1}{2}} r(Y_0)^{3^l} \tag{5}$$

where H_l is the number of heads in the l-th layer, γ is a constant related to the weight norms, λ' is the Lipschitz constant of MLP and d is the feature dimension.

Because the $H_l\gamma\lambda'/\sqrt{d}$ are smaller than 1, feature diversity $r(Y_l)$ is decreasing as block depth l increases [9]. Furthermore, when the number of heads H_l in the l-th layer is compressed, the feature diversity is drastically reduced as shown in Figure 1a. We call this *deep feature collapse*. To alleviate this problem, we propose a cross-block skip augmentation.

2.3 Gradient Collapse

The MSA and MLP modules contain linear layers, and projection dimensions such as QKV and MLP dimension are generally compressed. We found that the gradient scale became unstable with different compression rates for each transformer block, causing gradient explosion during training. We call this *gradient collapse*. The cause of this phenomenon is shown in the following theorem, established in [33].

Theorem 2. *Given a Pre-LN transformer with L layers assuming an output feature $\|Y_L\|_2^2$ are (ϵ, δ)-bounded and a hidden dimension d_m is same as a feature dimension d, the gradient of the weights of the last layer with probaility at least $0.99 - \delta - \frac{\epsilon}{0.9+\epsilon}$ is bounded by dimension d_m, i.e.,*

$$\|\frac{\partial \mathcal{L}}{\partial W^{2,L}}\|_F \leq O(d_m\sqrt{\frac{\ln d_m}{L}})$$

where L denotes number of blocks, and ϵ and $\delta = exp(-d\epsilon^2/8)$ are small numbers.

From Theorem 2, we can see that the scale of the Pre-LN transformer gradient is proportional to the dimension d_m being compressed. To alleviate this problem, we propose inserting additional layer normalization after the MSA and MLP layers. The approach, called Res-Post-LN, has been introduced [35] but has not been proven theoretically for effectiveness. Figure 1b shows that the gradient scale of Pre-LN becomes unstable with the compression rate. The instability of the gradient scale for each block causes gradient exploding. However, the gradient scale of Res-Post-LN is stable in each block. We demonstrate how Res-Post-LN prevents *gradient collapse* in Section 3.3.

3 Proposed Method

In this section, we introduce a search framework for exploring optimal architecture via the proposed BO process. The architecture domain is highly complex, and searching compression and architecture jointly is a high-dimensional problem that makes optimization procedures difficult to realize. To solve this issue,

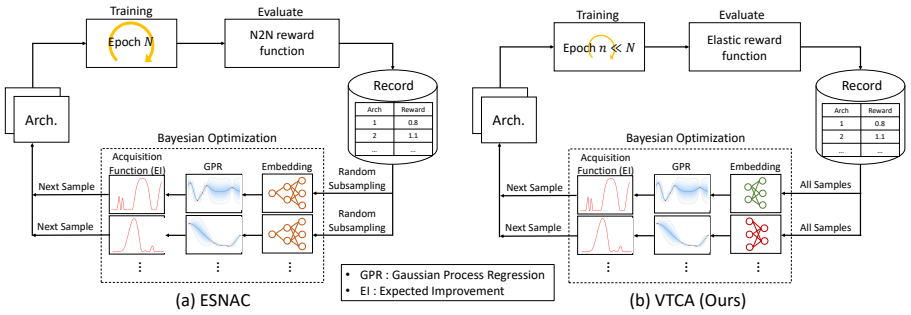


Fig. 3: Comparison of BO-based (a) ESNAC and (b) VTCA search process.

we propose our VTCA to extend ESNAC [5] which achieves compression for CNN in the embedding space. ESNAC can only be used with CNN and is difficult to apply to a large-scale dataset because of its high search cost. We adopt training speed estimation for a large-scale dataset and propose a novel elastic reward function that can represent a generalized manifold of rewards including naive and N2N rewards [1]. We compare the loops of ESNAC and VTCA in Figure 3.

3.1 Search Process with BO

The goal of the proposed method is to search a student transformer based on a given teacher network, maximizing the compression rate of weight parameters while still obtaining performance comparable to the teacher network. Formally, we aim to solve the following optimization problem:

$$x = \underset{x \in \mathcal{X}}{\operatorname{argmin}}\, f(x), \tag{6}$$

where $\mathcal{X}$ denotes the domain of transformer architectures and the function $f(x) : \mathcal{X} \mapsto \mathbb{R}$ evaluates a reward for how well our criterion is satisfied. Instead of an N2N reward [1], we propose an elastic reward function that controls the compression rate–performance trade-off.

The problem becomes hard to tackle because there is no certain form of f, a consequence of the complex relationship between compressed architecture and the corresponding reward. We adopt a BO approach, which is promising for optimizing expensive black-box functions. During the BO process, we denote the sampled architecture in the t-th round as $x_{1:t}$. The evaluated architectures for times 1 through t are denoted as $x_{1:t}$. The samples can be modeled as Gaussian processes (GP) and can be defined as follows:

$$f(x_{1:t}) \sim \mathcal{N}(\mu(x_{1:t}), \mathcal{K}(x_{1:t}, x_{1:t})) \tag{7}$$

where μ is the mean function; and $\mathcal{K}(x_{1:t}, x_{1:t})$ is the variance matrix. Then, the joint distribution of the preceding evaluated architectures $f(x_{1:t})$ and the next

compressed architecture $f(x_{t+1})$ can be represented by

$$\begin{bmatrix} f(x_{1:t}) \\ f(x_{t+1}) \end{bmatrix} \sim \begin{pmatrix} \mu(x_{1:t}) \\ \mu(x_{t+1}), \end{pmatrix} \begin{bmatrix} \mathcal{K}(x_{1:t}, x_{1:t}), & \mathbf{k}(x_{1:t}, x_{t+1}) \\ \mathbf{k}(x_{t+1}, x_{1:t}), & k(x_{t+1}, x_{t+1}) \end{bmatrix} \Bigg) \tag{8}$$

and the posterior predictive distibution of the next sample can be given by

$$\begin{aligned} f(x_{t+1}) &\sim \mathcal{N}(\mu(x_{t+1}), \sigma(x_{t+1})) \\ \mu(x_{t+1}) &= \mathbf{k}(x_{t+1}, x_{1:t})\mathcal{K}(x_{1:t}, x_{1:t})^{-1} f(x_{1:t}) \\ \sigma(x_{t+1}) &= k(x_{t+1}, x_{t+1}) - \mathbf{k}(x_{t+1}, x_{1:t})\mathcal{K}(x_{1:t}, x_{1:t})\mathbf{k}(x_{1:t}, x_{t+1}) \end{aligned} \tag{9}$$

The mean μ and variance σ of the unexplored architecture x_{t+1} can be calculated via the historic architectures.

We obtain the next architecture x_{t+1} using the expected improvement (EI) acquisition function [22,4]. The EI recommends a next sample that is most likely to maximize the objective function over current evaluated architectures:

$$\mathrm{EI}_t(x) = \mathbb{E}_t(\max(f(x) - f^*(x), 0)) \tag{10}$$

where $\mathbb{E}_t$ denotes the expectation over the posterior distribution at step t and $f^*(x)$ is the maximum value among evaluated architectures $f(x_t)$. The above algorithm is repeated up to a predefined step, and the architecture yielding the maximum value is returned.

3.2 Latent Embedding Space Search

The search space required to explore compression and architecture search jointly is very high-dimensional. High-dimensional BO suffers from drawbacks due to the curse of dimensionality. To address this issue, we adopt an architecture embedding function $h(\cdot;\theta)$ to map the compressed architecture to the embedding space according to the configuration parameters [5]. Here, θ represents the weight parameters for learning in the embedding function. We define the kernel function $k(x, x^{'};\theta)$ using an radial basis function (RBF) kernel:

$$k(x, x^{'};\theta) = \exp\Big(- \frac{\|h(x;\theta) - h(x^{'};\theta)\|^2}{2\sigma^2} \Big) \tag{11}$$

where σ is a hyperparameter and $h(\cdot,\theta)$ represents an embedding space for the high-dimensional architecture configuration. The functions $h(\cdot,\theta)$ and $k(x, x^{'};\theta)$ share the same weights θ. In what follows, we present the embedding function $h(\cdot,\theta)$ and describe how θ is learned during the search process.

The architecture embedding function $h(\cdot,\theta)$ needs to represent diverse compressed transformers adequately. In addition, it needs to be flexible enough to represent the inter- and intra-blocks relationships in the order of the transformer blocks. Therefore, we adopt a structured block correlation with two Bi-directional LSTMs motivated by [31]. One Bi-LSTM learns the relationships among the intra-block configuration information, the other Bi-LSTM among the

inter-block configuration information. After passing the Bi-LSTMs, we concatenate all the hidden states, applying L2 normalization to these states to obtain the embedding vector.

During the search stage, the weights θ are determined. The weights are trained to minimize the negative log posterior probability:

$$\mathcal{L}(\theta) = -\frac{1}{|D|} \sum_{i:x_i \in D} \log p(f(x_i)|f(D\backslash x_i); \theta) \tag{12}$$

where \denotes relative complement; and $f(D\backslash x_i) = [f(x_1), \ldots, f(x_{i-1}), f(x_{i+1}), \ldots, f(x_t)]$. Based on $k(\cdot, \cdot; \theta)$, the mean and covariance matrix of $p(f(x_i)|f(D\backslash x_i; \theta)$, which is a Gaussian distribution, can be calculated analytically [5].

Multiple Kernel and Dimension Strategy We adopt a multiple kernel strategy with diverse hidden dimensions per kernel. ESNAC trains a single model with different subsets of D instead of the entire evaluated architectures to avoid overfitting. The subset approach is helpful for small-scale datasets; however, a vision transformer that learns large-scale datasets is often unsuitable for evaluating architectures because of the large computation cost. Therefore, we determine the different dimensions of hidden states in the embedding function per kernel, allowing us to explore diverse architectures.

Training Speed Estimation We employ training speed estimation (TSE) [25] because the full training of each architecture of the vision transformer is expensive. Some studies have shown a correlation between training speed and generalization performance [25,12,20]. TSE estimates generalization performance with far fewer epochs n than the full number of epochs N. However, we define a reward that considers the compression rate with the TSE for architecture compression. To define this reward, the TSE needs to be expressed as an upper bound score, such as a compression rate between 0 and 1, regardless of the loss function. Therefore, we propose a normalized cross-entropy score (NCE score) that extends NCE [21] during the search process:

$$NCE_{score} = K \cdot \frac{-\sum_{k=1}^{K} q(k|i) \log p(k|i)}{-\sum_{j=1}^{K} \sum_{k=1}^{K} q(y=j|i) \log p(k|i)} \tag{13}$$

where K is the number of classes, and i denotes input images. The numerator is the cross-entropy (CE) loss, and the denominator is the sum of the CEs for each class. Then, $NCE_{score} \in (0, 1)$ can be represented as upper-bounded scores to evaluate rewards.

Elastic Reward Function We introduce an elastic reward function including compression rate and accuracy (NCE score) to evaluate compressed architectures. The N2N reward [1] was proposed as an alternative to a naïve reward to maximize compression rate while preserving high accuracy. However, this reward

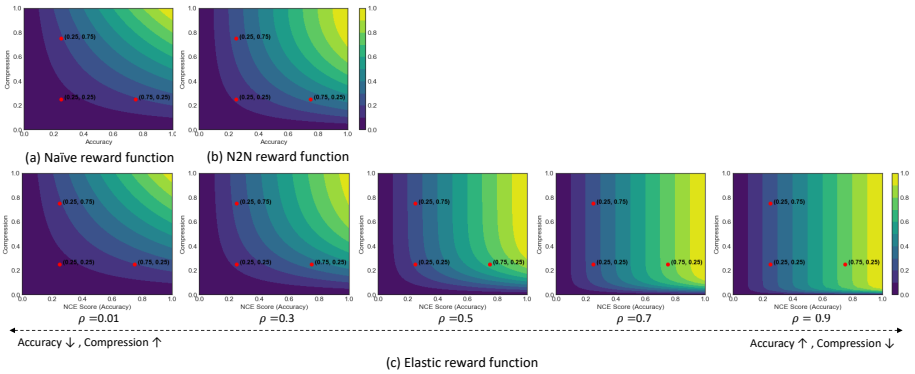


Fig. 4: Manifold of reward functions : (a) naive reward, (b) N2N reward (c) elastic reward function.

cannot control the penalty between accuracy and compression rate. The elastic reward function is motivated by the sigmoid function and can be represented as a generalized function that incorporates naive and N2N reward functions [1] by adjusting the scale factor ρ. Figure 4 shows the manifold of the elastic reward functions. The scale factor ρ controls the trade-off between compression rate and accuracy. The larger the scale factor, the more the compression rate is penalized (↑ accuracy and ↓ compression). The smaller the scale factor, conversely, the more the accuracy is penalized (↓ accuracy and ↑ compression). The elastic reward function is defined as follows:

$$f(x) = \left(\frac{2}{1 + \exp(-10^{2\rho} \cdot C(x))} - 1 \right) \cdot A(x)$$

where $C(x)$ is the compression rate, $A(x)$ is the accuracy (NCE score), and ρ is the scale factor.

Training Loss An architecture with optimal reward is trained with the following objective function [36]:

$$\min_{W,gt} \mathcal{L}(W, gt) = \ell(W, gt) + \lambda_l \ell_{distill}(W, W_t) \tag{14}$$

where $\ell(\cdot,\cdot)$ is cross-entropy loss and $\ell_{distill}(\cdot,\cdot)$ is knowledge distillation loss, namely KL-divergence between the compressed and teacher networks. W_t denotes weights for the uncompressed teacher network, and λ_l denotes the hyper-parameter for scale of loss.

3.3 Search Space

We define the search space based on the teacher transformer. The search space is constructed from all the architectures that can be obtained by manipulating

the teacher network with the following three operations: (1) block-wise removal and shrinkage, (2) cross-block skip augmentation and (3) addition of res-post layer normalization.

Block-wise Removal and Shrinkage Since we jointly compress and search the student architecture from the given teacher architecture, we only consider making architectures smaller than the given network. Block-wise removal refers to dropping the transformer block; block-wise shrinkage refers to compressing the number of heads H_l and the hidden dimension of MLP in the l-th layer.

Cross-block Skip Augmentation The addition of cross-block skip augmentation is employed to prevent *deep feature collapse* as described in Theorem 1. Cross-block skip augmentation connected from the k-th layer to the l-th layer can be formulated as:

$$\text{SkipAug}(Y_l) = Y_l + T_{lk}(Y_k; \Theta_{lk}) \tag{15}$$

where $T_{lk}(\cdot)$ is the augmentation operation from the k-th layer to the l-th layer and $\Theta_{lk} \in \mathbb{R}^{d \times d}$ denotes the weight matrix. The augmentation block consists of linear projection, an activation function (e.g., GELU), and layer normalization and can be defined as:

$$T_{lk}(Y_k; \Theta_{lk}) = \mathbb{I}(b_{lk})\text{LN}(\sigma(Y_k\Theta_{lk})), \text{ where } \mathbb{I}(b_{lk}) = \begin{cases} 1, & b_{lk} \in B \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

The $\mathbb{I}(b_{lk})$ is an indicator function for the cross-block skip augmentation set B, and the b_{lk} denotes skip augmentation blocks connected from the k-th layer to the l-th layer.

We analyze how cross-block skip augmentation prevents *deep feature collapse* in the following theorem.

Theorem 3. *Given a model with the cross-block skip augmentation, the diversity $r(Y_l)$ of features in the l-th layer can be bounded by that of the input data $r(Y_0)$:*

$$r(Y_l) \leq \left(\frac{4H_l\gamma\lambda'}{\sqrt{d}}\right)^{\frac{3^l-1}{2}} r(Y_0)^{3^l} + \underbrace{\mathbb{I}(b_{lk})\alpha_{lk}r(Y_k)}_{\geq 0}$$

where $\alpha_{lk} = \lambda_{LN}\lambda_a\|\Theta_{lk}\|$. Here, Θ_{lk} is the the weight matrix in the augmentation block from the k-th layer to the l-th layer, λ_{LN}, λ_a are the Lipschitz constants of layer normalization $LN(\cdot)$ and non-linear activation function $\sigma(\cdot)$ respectively, and $\mathbb{I}(b_{lk})$ is the indicator function defined in equation 16.

Comparing with Theorem 1, the cross-block skip augmentation introduces an additional term $\mathbb{I}(b_{lk})\alpha_{lk}r(Y_k)$, which is greater than or equal to zero and prevents the feature diversity from decreasing doubly exponentially. A detailed proof of Theorem 3 is given in Appendix A.1.

Adding Res-Post Layer Normalization To prevent *gradient collapse*, we apply layer normalization at the end of each residual block as shown in Figure 2. This is referred to as Res-Post-LN in this paper. Res-Post-LN has been proposed in [35], however its effectiveness has not been theoretically proven. We analyze how Res-Post-LN prevents *gradient collapse* in the subsequent theorem.

Theorem 4. *Given a Res-Post-LN transformer with L layers assuming the an output feature* $\|Y_L\|_2^2$ *are* (ϵ, δ)*-bounded and a hidden dimension* d_m *is same as a feature dimension* d*, the gradient of the weights of the last layer with probaility at least* $0.99 - \delta - \frac{\epsilon}{0.9+\epsilon}$ *is bounded by dimension* d_m*, i.e.,*

$$\|\frac{\partial \mathcal{L}}{\partial W^{2,L}}\|_F \leq O(\sqrt{\frac{d_m \ln d_m}{L}})$$

where L denotes number of blocks, and ϵ *and* $\delta = exp(-d\epsilon^2/8)$ *are small numbers.*

From Theorem 4, the scale of gradient for the Res-Post-LN transformer is less affected by the feature dimension d_m. As shown in Figure 1b, the Pre-LN gradient expectation is fluctuates for each layer L according to the compression rate. However, the gradient expectation of Res-Post-LN is stable regardless of the compression rate. The stability of the gradient size for each block ensures stable learning. A detailed proof of Theorem 4 can be found in the supplementary material.

Representation for Block Configurations The representation for each block configuration is defined by a vector of length $(2n + 3)$, where n is the maximum number of blocks in teacher network. The $2n$ dimensions encode a directed acyclic graph for cross-block skip augmentation. The first n-dimensions represent the input from another node, while the remaining n-dimensions represent the output from each node. The attribute of each block is denoted by three numbers: the number of heads, hidden dimensions of MLP, and drop block.

4 Experiments

We evaluate the VTCA method for image classifcation on the ImageNet challenge dataset [17]. We implement experiments for VTCA on DeiT-Tiny/Small/Base [29], comparing the automatically found compressed architectures to recent compression methods. We compare the compression rate and performance with varying scale factors ρ. We also perform an ablation study on how each architecture module affects the performance results.

4.1 Comparison Results

The experiment results are in Table 1. Here, the VTCA results are all obtained experimentally with a scale factor of 0.7. Our VTCA method achieves competitive accuracies compared with recent methods. We adopt several of the latest

Table 1: Comparison on ImageNet of vision transformer compressed by VTCA with other competitive methods.

Model	Method	Top-1 Acc (%)	FLOPs (G)	Design Type
DeiT-Tiny	Baseline [29]	72.2	1.3	-
	HVT [24]	69.64 (-2.56)	0.64	Patch Reduction
	SViTE [7]	70.12 (-2.08)	0.99	Pruning
	UVC [36]	71.8 (-0.4)	0.69	Pruning
	VTCA	71.63 (-0.57)	0.99	BO
DeiT-Small	Baseline [29]	79.8	4.6	-
	PatchSlimming [26]	79.4 (-0.4)	2.6	Patch Reduction
	IA-RED^2 [23]	79.1 (-0.7)	-	Patch Reduction
	PoWER [36]	78.3 (-1.5)	2.7	Patch Reduction
	HVT [24]	78.0 (-1.8)	2.4	Patch Reduction
	SViTE [7]	79.22 (-0.58)	3.14	Pruning
	SCOP [36]	77.5 (-2.3)	2.6	Pruning
	UVC [36]	79.44 (-0.36)	2.65	Pruning
	VTCA	79.45 (-0.35)	3.11	BO
DeiT-Base	Baseline [29]	81.8	17.6	-
	PatchSlimming [26]	81.5 (-0.3)	9.8	Patch Reduction
	IA-RED^2 [23]	80.9 (-0.9)	11.8	Patch Reduction
	VTP [37]	80.7 (-1.1)	10.0	Pruning
	UVC [36]	80.57 (-1.23)	8.0	Pruning
	VTCA	81.94 (+0.14)	11.9	BO

patch reductions, specifically PoWER [11], HVT [24], PatchiSlimming [26], and IA-RED^2 [23], as well as pruning methods, namely SCOP [28], VTP [37], SViTE [7], and UVC [36].

VTCA avoids accuracy losses compared to pruning and patch reduction methods. On DeiT-Tiny/Small it performs competitively on accuracy compared to UVC, the latest pruning method. In particular, VTCA on DeiT-Base achieves better accuracy than baseline while decreasing FLOPs. VTCA obtains 81.94% of Top-1 accuracy while FLOPs are comparable to IA-RED.^{2}. We observe that VTCA shows competitive accuracy, but FLOPs are higher than with the pruning method. This is due to slightly increased FLOPs with the addition of cross-block skip augmentation and layer normalization.

4.2 Effect of Scale Factor

We performed experiments on how the change of the scale factor ρ in our elastic reward function affects accuracy and compression rate, setting the scale factor ρ to 0.3, 0.5, 0.7, and 0.9; the results are shown in Table 2. Because accuracy becomes more important as the scale factor increases, we see an accuracy gain but with loss of FLOPs. Conversely, when the scale factor is decreased, we obtain a FLOPs gain; but performance losses.

4.3 Ablation Study

As VTCA foregrounds integrating compression and architecture search simultaneously, it is natural to question how each module contributes to the final

Table 2: Changes in accuracy and FLOPs according to scale factor.

Model	Scale factor	Top-1 Acc. (%)	FLOPs (G)
DeiT-Tiny	$\rho = 0.9$	71.73	1.03
	$\rho = 0.7$	71.63	0.99
	$\rho = 0.5$	70.23	0.89
	$\rho = 0.3$	68.8	0.72

result. We conducted an ablation study by removing each module; the results are shown in Table 3. The effectiveness of knowledge distillation has already been demonstrated in [36], but an ablation study on knowledge distillation was not performed.

Table 3: Ablation study on the modules implemented on VTCA.

Method	Top-1 Acc. (%)	FLOPs (G)
Uncompressed baseline	72.2	1.3
Compression Only	69.78	0.84
Compression With Res-Post-LN Only	71.16	0.95
Compression With Skip Augmentation Only	70.89	0.91
VTCA-tiny	71.63	0.99

We first determined the result when we conduct only block-wise removal and shrink in our method, demonstrating that only performing compression will significantly impair accuracy on DeiT-Tiny by over 2.4%. That is expected, as simply compressing the model architecture makes it very unstable and prone to collapse.

We then conducted the experiment with Res-Post-LN only. This implies integrating only compression and Res-Post-LN with knowledge distillation. Better performance was achieved than with compression alone. Much better accuracy is obtained than with cross-block skip augmentation, owing to preventing *gradient collapse* and ensuring stable training.

When the experiment was performed with cross-block skip augmentation only, we observed that gradient exploding occurs in some architectures during the search process. This made it difficult to find an optimal architecture, resulting in performance loss; (e.g., approximately 1.4% drop on DeiT-Tiny). Overall, the results of our ablation studies support the effectiveness of optimizing compression and architecture search jointly.

5 Conclusion

In this paper, we propose a VTCA that jointly compresses and searches the architecture of a vision transformer. We consider *deep feature collapse*, in which the number of heads is compressed and the feature diversity rapidly decreases as

the layer becomes deeper, and *gradient collapse*, in which the scale of the gradient changes rapidly for each layer as the weight dimensions compress. To alleviate this problem, we propose a cross-block skip augmentation to prevent *feature collapse* and Res-Post-LN architecture to prevent *gradient collapse*. Experiments demonstrate that VTCA achieves competitive performance compared to recent patch reduction and pruning methods. Our future work will extend VTCA to achieve as many FLOPs as pruning without accuracy loss.

Acknowledgements This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2020R1I1A3074639) and the Technology Innovation Program (20011875, Development of AI based diagnostic technology for medical imaging devices) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea).

References

1. Ashok, A., Rhinehart, N., Beainy, F., Kitani, K.M.: N2n learning: Network to network compression via policy gradient reinforcement learning. arXiv preprint arXiv:1709.06030 (2017)
2. Aziznejad, S., Gupta, H., Campos, J., Unser, M.: Deep neural networks with trainable activations and controlled lipschitz constant. IEEE Transactions on Signal Processing **68**, 4688–4699 (2020)
3. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
4. Brochu, E., Cora, V.M., De Freitas, N.: A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599 (2010)
5. Cao, S., Wang, X., Kitani, K.M.: Learnable embedding space for efficient neural architecture compression. arXiv preprint arXiv:1902.00383 (2019)
6. Chen, M., Peng, H., Fu, J., Ling, H.: Autoformer: Searching transformers for visual recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12270–12280 (2021)
7. Chen, T., Cheng, Y., Gan, Z., Yuan, L., Zhang, L., Wang, Z.: Chasing sparsity in vision transformers: An end-to-end exploration. Advances in Neural Information Processing Systems **34**, 19974–19988 (2021)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Dong, Y., Cordonnier, J.B., Loukas, A.: Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In: International Conference on Machine Learning. pp. 2793–2803. PMLR (2021)
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)

11. Goyal, S., Choudhury, A.R., Raje, S., Chakaravarthy, V., Sabharwal, Y., Verma, A.: Power-bert: Accelerating bert inference via progressive word-vector elimination. In: International Conference on Machine Learning. pp. 3690–3699. PMLR (2020)
12. Hardt, M., Recht, B., Singer, Y.: Train faster, generalize better: Stability of stochastic gradient descent. In: International conference on machine learning. pp. 1225–1234. PMLR (2016)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.J., Han, S.: Amc: Automl for model compression and acceleration on mobile devices. In: Proceedings of the European conference on computer vision (ECCV). pp. 784–800 (2018)
15. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE international conference on computer vision. pp. 1389–1397 (2017)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
18. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation **1**(4), 541–551 (1989)
19. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE international conference on computer vision. pp. 2736–2744 (2017)
20. Lyle, C., Schut, L., Ru, R., Gal, Y., van der Wilk, M.: A bayesian perspective on training speed and model selection. Advances in Neural Information Processing Systems **33**, 10396–10408 (2020)
21. Ma, X., Huang, H., Wang, Y., Romano, S., Erfani, S., Bailey, J.: Normalized loss functions for deep learning with noisy labels. In: International conference on machine learning. pp. 6543–6553. PMLR (2020)
22. Mockus, J.B., Mockus, L.J.: Bayesian approach to global optimization and application to multiobjective and constrained problems. Journal of optimization theory and applications **70**(1), 157–172 (1991)
23. Pan, B., Panda, R., Jiang, Y., Wang, Z., Feris, R., Oliva, A.: Ia-red^2 : Interpretability-aware redundancy reduction for vision transformers. Advances in Neural Information Processing Systems **34**, 24898–24911 (2021)
24. Pan, Z., Zhuang, B., Liu, J., He, H., Cai, J.: Scalable vision transformers with hierarchical pooling. In: Proceedings of the IEEE/cvf international conference on computer vision. pp. 377–386 (2021)
25. Ru, B., Lyle, C., Schut, L., Fil, M., van der Wilk, M., Gal, Y.: Speedy performance estimation for neural architecture search. arXiv preprint arXiv:2006.04492 (2020)
26. Tang, Y., Han, K., Wang, Y., Xu, C., Guo, J., Xu, C., Tao, D.: Patch slimming for efficient vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12165–12174 (2022)
27. Tang, Y., Han, K., Xu, C., Xiao, A., Deng, Y., Xu, C., Wang, Y.: Augmented shortcuts for vision transformers. Advances in Neural Information Processing Systems **34** (2021)

28. Tang, Y., Wang, Y., Xu, Y., Tao, D., Xu, C., Xu, C., Xu, C.: Scop: Scientific control for reliable neural network pruning. Advances in Neural Information Processing Systems **33**, 10936–10947 (2020)
29. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. pp. 10347–10357. PMLR (2021)
30. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
31. Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., Cremers, D.: Image-based localization using lstms for structured feature correlation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 627–637 (2017)
32. Wood, G., Zhang, B.: Estimation of the lipschitz constant of a function. Journal of Global Optimization **8**(1), 91–103 (1996)
33. Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., Liu, T.: On layer normalization in the transformer architecture. In: International Conference on Machine Learning. pp. 10524–10533. PMLR (2020)
34. Yang, H., Yin, H., Molchanov, P., Li, H., Kautz, J.: Nvit: Vision transformer compression and parameter redistribution. arXiv preprint arXiv:2110.04869 (2021)
35. Yao, Z., Cao, Y., Lin, Y., Liu, Z., Zhang, Z., Hu, H.: Leveraging batch normalization for vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 413–422 (2021)
36. Yu, S., Chen, T., Shen, J., Yuan, H., Tan, J., Yang, S., Liu, J., Wang, Z.: Unified visual transformer compression. arXiv preprint arXiv:2203.08243 (2022)
37. Zhu, M., Tang, Y., Han, K.: Vision transformer pruning. arXiv preprint arXiv:2104.08500 (2021)

A Appendix

A.1 Proof of Theorem 3

Here, we prove Theorem 3 on how cross-block skip augmentation prevents feature collapse of vision transformers.

The feature diversity $r(T_{lk}(Y_l))$ outputted by the augmentation operation $T_{lk}(\cdot)$ can be bounded as:

$$\begin{aligned} r(T_{lk}(Y_k)) \leq \|T_{lk}(Y_k) - T_{lk}(\mathbf{1}\mathbf{y}_k^T)\| &= \|\mathbb{I}(b_{lk})\mathrm{LN}(\sigma(Y_k\Theta_{lk})) - \mathbb{I}(b_{lk})\mathrm{LN}(\sigma(\mathbf{1}\mathbf{y}_k^T\Theta_{lk}))\| \\ &= \|\mathbb{I}(b_{lk})[\mathrm{LN}(\sigma(Y_k\Theta_{lk})) - \mathrm{LN}(\sigma(\mathbf{1}\mathbf{y}_k^T\Theta_{lk}))]\| \end{aligned}$$

where the inequality comes from equation 4 defining the feature diversity. The b_{lk} denotes skip augmentation blocks connected from the k-th layer to the l-th layer. The $\mathbb{I}(b_{lk})$ is an indicator function for the cross-block skip augmentation set B defined in equation 16.

Using Lipschitz continuity [32,2] of the linear projection, a non-linear activation function, and layer normalization, the bound can be further described as:

$$r(T_{lk}(Y_k)) \leq \mathbb{I}(b_{lk})\lambda_{LN}\lambda_a\|\Theta_{lk}\|\|Y_k - \mathbf{1}\mathbf{y}_k^T\| = \mathbb{I}(b_{lk})\lambda_{LN}\lambda_a\|\Theta_{lk}\|r(Y_k)$$

where λ_{LN} and λ_a denotes the Lipschitz constant of layer normalization and the non-linear activation function respectively, and Θ_{lk} is the weight matrix. Combining with a multi-head attention module (Corollary 3.2 in [9]) and the cross-block skip augmentation, diversity after the SkipAug module is bounded as

$$\begin{aligned} r(Y_l) &\leq \left(\frac{4H\gamma\lambda'}{\sqrt{d}}\right)^{\frac{3^l-1}{2}} r(Y_0)^{3^l} + \mathbb{I}(b_{lk})\lambda_{LN}\lambda_a\|\Theta_{lk}\|r(Y_k) \\ &\leq \left(\frac{4H\gamma\lambda'}{\sqrt{d}}\right)^{\frac{3^l-1}{2}} r(Y_0)^{3^l} + \mathbb{I}(b_{lk})\alpha_{lk}r(Y_k) \end{aligned}$$

where $\alpha_{lk} = \lambda_{LN}\lambda_a\|\Theta_{lk}\|$, γ is a constant related to the weight norms and λ' is the Lipschitz constant of MLP. The above inequality corresponds to Theorem 3.