USDC: Unified Static and Dynamic Compression for Visual Transformer

Huan Yuan, Chao Liao, Jianchao Tan*, Peng Yao, Jiyuan Jia, Bin Chen, Chengru Song, Di Zhang
Kuaishou Technology

Visual Transformers have achieved great success in almost all vision tasks, such as classification, detection, and so on. However, the model complexity and the inference speed of the visual transformers hinder their deployments in industrial products. Various model compression techniques focus on directly compressing the visual transformers into a smaller one while maintaining the model performance, however, the performance drops dramatically when the compression ratio is large. Furthermore, several dynamic network techniques have also been applied to dynamically compress the visual transformers to obtain input-adaptive efficient sub-structures during the inference stage, which can achieve a better trade-off between the compression ratio and the model performance. The upper bound of memory of dynamic models is not reduced in the practical deployment since the whole original visual transformer model and the additional control gating modules should be loaded onto devices together for inference. To alleviate two disadvantages of two categories of methods, we propose to unify the static compression and dynamic compression techniques jointly to obtain an input-adaptive compressed model, which can further better balance the total compression ratios and the model performances. Moreover, in practical deployment, the batch sizes of the training and inference stage are usually different, which will cause the model inference performance to be worse than the model training performance, which is not touched by all previous dynamic network papers. We propose a sub-group gates augmentation technique to solve this performance drop problem. Extensive experiments demonstrate the superiority of our method on various baseline visual transformers such as DeiT, T2T-ViT, and so on.

1. Introduction

Recently, transformer-based models [1, 2] have achieved significant performance improvement in both natural language processing [1, 3] and computer vision [2, 4]. Based on the self-attention mechanism, the ViT model [2, 5] and the subsequent improved ViT model have gradually surpassed CNN-based models in image classification tasks. However, the ViT-based model is gradually becoming deeper and more complex, the huge computing resources and the large parameters of the ViT model will hinder its deployment and application in real-world scenarios. Therefore, many existing works are devoted to slimming and accelerating ViT models using model compression techniques to reduce the cost of deployment.

The existing compression methods for ViT-based models can be divided into static compression [6–10] and dynamic compression [2, 11–14]. Static compression methods attempt to permanently and explicitly remove the sub-structure of the network for high efficiency without compromising the performance of the network. Because the complexity of the ViT model is proportional to the number of input tokens, Patch Slimming [7] proposed to reduce the computational cost of the ViT models by removing redundant input tokens. UVC [8] proposed to formulate the weight learning, pruning, and layer skipping as a joint optimization problem, thereby compressing the ViT model from multiple dimensions and obtaining minimal performance loss. Although the static compression method adopts various strategies to reduce the loss of performance while compressing the model,

*Corresponding Author.

Preprint. Work in progress.

model and has always been used to reduce the number of parameters and computation of DNN models [8, 37, 38]. According to the granularity of pruning, static compression can be divided into two types: unstructured pruning [39, 40] and structured pruning [8, 37, 41–44]. Unstructured pruning methods sparsify the elementwise weight values of the model, which can reduce the storage capacity of the model, but it is difficult for most of the current hardware to support the reasoning of this sparse compressed model. Structured pruning usually takes channels [37, 44, 45], layers, blocks [8], etc. as a unit, and structurally removes the entire unit. Structured pruning is widely used in DNN because it can slim down the model well and is hardware-friendly. Structured pruning, through the basic pruning unit, is one or more Channels of filters or weight matrices, can slim down the model, and also has hardware-friendly features. The importance or magnitude of each channel of the conv-layer can be associated with the batch norm layer[44, 46]. [47] use the corresponding scaling factor in the batch norm layer after a channel as the importance of the channel. Recently, there have been more and more static structured pruning methods for ViT models. Patch slimming [7] reduces the input of the model by removing redundant input tokens. UVC [8] optimizes the structure of ViT models through the dual optimizer method to obtain a lightweight model. [9] apply multi-dimensional compression on the ViT models through guided Gaussian process search. The complexity of the ViT model makes it have better performance, but the high storage space and computing resource consumption are important reasons for it to be difficult to effectively apply on various hardware platforms [48, 49].

3. Approach

3.1. ViT Preliminaries

Vision Transformers [5, 18, 50] split the input images into raw patches and embed these fixed-size patches into tokens by patch embeddings and positional embeddings, and then those tokens will be feed into sequential transformer encoder layers. Similar to transformers [1, 3] in NLP, the encoder layers of ViT models usually consist of multi-head self-attention (MHSA) block and feed-forward network (FFN) block. MHSA block combines the output self-attentions information of heads, every single head $\mathrm{head}_{i,l}$ project all input tokens $\boldsymbol{Z}_\ell$ into three matrices by three linear layers, named $\boldsymbol{Q}_{i,\ell}$, $\boldsymbol{K}_{i,\ell}$ and $\boldsymbol{V}_{i,\ell}$, where l and i denote the l_{th} encoder of ViT and i_{th} head of MHSA.

$$\begin{aligned} \boldsymbol{Q}_{i,\ell} &= \boldsymbol{Z}_\ell \boldsymbol{W}_{i,\ell}^{Q} \\ \boldsymbol{K}_{i,\ell} &= \boldsymbol{Z}_\ell \boldsymbol{W}_{i,\ell}^{K} \\ \boldsymbol{V}_{i,\ell} &= \boldsymbol{Z}_\ell \boldsymbol{W}_{i,\ell}^{V} \end{aligned} \tag{1}$$

where $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$ and $\boldsymbol{W}^V$ denote the trainable parameters of three linear layer. Then these three matrices will be used to calculate the self-attention as shown in Eq. 2:

$$\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}\right)\boldsymbol{V} \tag{2}$$

Finally, MHSA concatenates the result from every head and projects them as the output by linear layer. FFN block takes the output token of MHSA as input and then projects it by two-layer linear: $\{\boldsymbol{W}_\ell^I, \boldsymbol{W}_\ell^O\}$. For simplicity, in all equations, the bias of the linear layers is not mentioned.

$$\begin{aligned} \mathrm{head}_{i,\ell} &= \mathrm{Attn}\left(\boldsymbol{Z}_\ell \boldsymbol{W}_{i,\ell}^{Q}, \boldsymbol{Z}_\ell \boldsymbol{W}_{i,\ell}^{K}, \boldsymbol{Z}_\ell \boldsymbol{W}_{i,\ell}^{V}\right) \\ \mathrm{MHSA}\left(\boldsymbol{Z}_\ell\right) &= \mathrm{Concat}\left(\mathrm{head}_{1,\ell}, \ldots, \mathrm{head}_{H,\ell}\right) \boldsymbol{W}_\ell^O \\ \mathrm{FFN}\left(\boldsymbol{Z}'_\ell\right) &= LN\left(\boldsymbol{Z}'_\ell\right) \boldsymbol{W}_\ell^I \boldsymbol{W}_\ell^O \end{aligned} \tag{3}$$

The LayerNorm [51] and residual connections are applied on both MHSA and FFN blocks, then the output of ℓ_{th} layer is defined as:

$$\begin{aligned} \boldsymbol{Z}'_\ell &= \mathrm{MHSA}\left(\mathrm{LN}\left(\boldsymbol{Z}_{\ell-1}\right)\right) + \boldsymbol{Z}_{\ell-1}, \\ \boldsymbol{Z}_\ell &= \mathrm{FFN}\left(\mathrm{LN}\left(\boldsymbol{Z}'_\ell\right)\right) + \boldsymbol{Z}'_\ell, \end{aligned} \tag{4}$$

The class token of the last transformer encoder layer will be fed into a linear layer, and the output of this linear layer will be used for final classification prediction.

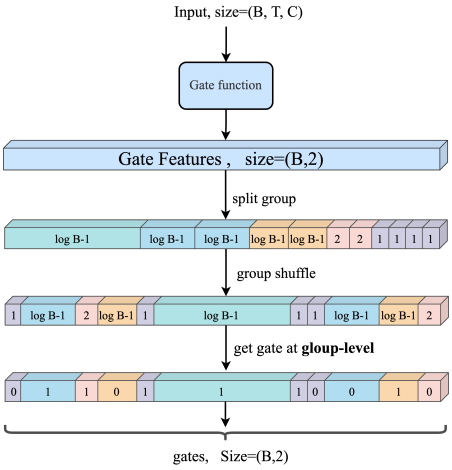

Figure 2: Group-level dynamic gates augmentation.

Similar to dynamic gate network searching, we use a differentiable architecture method [52] to optimize static compression parameters. Under static compression training, the MHSA and the FFN in ViT models defined in Eq. 3 would be calculated as follows:

$$\begin{aligned}
&\text{head}'_{\ell,i} = \boldsymbol{\alpha}^{h}_{\ell,i,0}\,\text{Attn}\left(\boldsymbol{Q}_{\ell,i}, \boldsymbol{K}_{\ell,i}, \boldsymbol{V}_{\ell,i}\right) \\
&\text{MHSA}'\left(\boldsymbol{Z}_{\ell}\right) = \text{Concat}\left(\text{head}'_{\ell,1}, \ldots, \text{head}'_{\ell,H}\right) \boldsymbol{W}^{O}_{\ell} \\
&\text{FFN}'\left(\boldsymbol{Z}'_{\ell}\right) = LN\left(\boldsymbol{Z}'_{\ell}\right) \boldsymbol{W}^{I}_{\ell} \boldsymbol{\alpha}^{n}_{\ell,:,0} \boldsymbol{W}^{O}_{\ell}
\end{aligned} \tag{7}$$

Meanwhile, the static compression parameters $\{\boldsymbol{\alpha}^{a}_{\ell} \in \mathbf{R}^{1\times 2}\}$ are for MHSA block and $\{\boldsymbol{\alpha}^{f}_{\ell} \in \mathbf{R}^{1\times 2}\}$ are for FFN block in ℓ_{th} transformer encoder layer. We optimize the dynamic compression and static compression jointly, therefore, the dynamic gates $\mathbf{g}$ and static compression parameters $\boldsymbol{\alpha}$ will be calculated and trained together, the block defined in Eq. 5 becomes:

$$\begin{aligned}
&\boldsymbol{Z}'_{\ell} = \text{MHSA}'\left(\text{LN}\left(\boldsymbol{Z}_{\ell-1}\right)\right) + \boldsymbol{Z}_{\ell-1}, \\
&\boldsymbol{Z}''_{\ell} = \mathbf{g}_{\ell,0} \boldsymbol{\alpha}^{a}_{\ell,0} \boldsymbol{Z}'_{\ell} + \left(1 - \mathbf{g}_{\ell,0}\right) \boldsymbol{\alpha}^{a}_{\ell,1} \boldsymbol{Z}_{\ell-1}, \\
&\boldsymbol{Z}'''_{\ell} = \text{FFN}'\left(\text{LN}\left(\boldsymbol{Z}''_{\ell}\right)\right) + \boldsymbol{Z}''_{\ell} \\
&\boldsymbol{Z}_{\ell} = \mathbf{g}_{\ell,1} \boldsymbol{\alpha}^{m}_{\ell,0} \boldsymbol{Z}'''_{\ell} + \left(1 - \mathbf{g}_{\ell,1}\right) \boldsymbol{\alpha}^{m}_{\ell,1} \boldsymbol{Z}''_{\ell},
\end{aligned} \tag{8}$$

After static compression, the structure (head, channel, and block) with static parameters meet the conditions of $(\boldsymbol{\alpha}_{:,0} < \boldsymbol{\alpha}_{:,1})$ would be pruned. The retained structure will continue to be optimized together with the dynamic compression, see the description below for details.

3.4. Unified Static and Dynamic Compression

Static compression can explicitly prune the sub-structure of the ViT models, thereby reducing the number of parameters and computation of the ViT models. However, when the retained ratio of the static compression pruned model is too small, it will reduce the representation ability and robustness of the ViT models. Dynamic compression can dynamically execute the network according to the input, but the dynamic decision networks are usually used as the controller, which increases the upper bound of backbone models and inference time. To balance the advantages and shortcomings of these two methods, we propose USDC, a joint compression mechanism that unifies static and dynamic compression together. We illustrate three types of compression techniques in Figure 3.

The optimization of USDC is divided into two stages. In the first stage, we use the differentiable neural search method [52] to optimize the static compression parameters to get lightweight ViT models and to search dynamic gates network as described above at Sec. 3.2. We jointly optimize

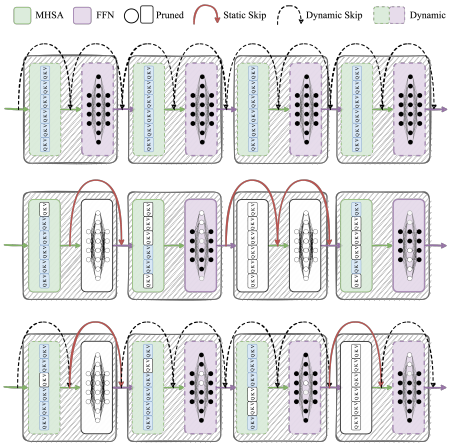

Figure 3: An illustration of the optimization results from three types of compression techniques. When reaching the same compression ratio, joint compression will automatically choose optimal dynamic and static percentile schedules to balance the performance and compression ratios.

the cross-entropy loss $\mathcal{L}_{cls}$ of the image classification task and compression resource loss $\mathcal{L}_{res}$. We design the resource loss $\mathcal{L}_{res}$ as the square of the FLOPs difference between the compressing model and source backbone model. As shown in Eq. 9, $\mathcal{F}(\boldsymbol{\alpha}; \mathbf{g})$ represents the FLOPs of the whole ViT model during the optimization of USDC.

$$\begin{aligned} \mathcal{L}_{total} &= \mathcal{L}_{cls} + \gamma \mathcal{L}_{res} \\ \mathcal{L}_{res} &= \left(\mathcal{F}(\boldsymbol{\alpha}; \mathbf{g}) - f_t\right)^2 \end{aligned} \tag{9}$$

where f_t is the target compression ratio and γ is the hyperparameter to balance task loss and resource loss, the value of $\mathcal{F}(\boldsymbol{\alpha}; \mathbf{g})$ is affected by static compression parameters $\boldsymbol{\alpha}$ and dynamic compression gates $\mathbf{g}$ as below:

$$\begin{aligned} \mathrm{F}'_{\text{attn}} &= \boldsymbol{\alpha}^a_{\ell,0} \sum_{h \in H} \boldsymbol{\alpha}^h_{\ell,i,0}\, \mathrm{F}_{\text{attn}} \\ \mathrm{F}'_{\text{ffn}} &= \boldsymbol{\alpha}^f_{\ell,0} \sum_{h \in H} \boldsymbol{\alpha}^n_{\ell,i,1}\, \mathrm{F}_{\text{ffn}} \\ \mathcal{F}(\boldsymbol{\alpha}; \mathbf{g}) &= \sum_{l \in \mathbf{L}} \left(\mathbf{g}_{\ell,0}\, \mathrm{F}'_{\text{attn}} + \mathbf{g}_{\ell,1}\, \mathrm{F}'_{\text{ffn}} + \sum_{k \in K} \boldsymbol{\alpha}^k_{\ell,k}\, \mathrm{F}_{\boldsymbol{G},\ell,k} \right) + \mathrm{F}_o \end{aligned} \tag{10}$$

Where $\mathrm{F}'_{\text{attn}}$ and F'_{ffn} are the FLOPs scales of MHSA and FFN block which are normalized by total FLOPs of the model, F_o is the FLOPs scale of both embedding layers and final classifier layer in ViT model. $\mathrm{F}_{\boldsymbol{G},\ell,:}$ are the FLOPs scale of dynamic gates network space of ℓ_{th} encoder layer.

In the second stage, according to the optimizing results of the static compression parameters $\boldsymbol{\alpha}$ on the first stage, **we explicitly prune the ViT model and continue to fine-tune the pruned model under dynamic compression constraint**. In the second stage, each transformer encoder layer selects a fixed $\boldsymbol{G}_\ell$ with the largest value in $\boldsymbol{\alpha}^{\boldsymbol{g}}_{\boldsymbol{\ell}}$. The FLOPs $\mathcal{F}(\mathbf{g})$ of ViT model on second stage is:

$$\mathcal{F}(\mathbf{g}) = \sum_{l \in \mathbf{L}} \left(\mathbf{g}_{\ell,0}\, \mathrm{F}''_{\text{attn}} + \mathbf{g}_{\ell,1}\, \mathrm{F}''_{\text{ffn}} + \mathrm{F}_{\boldsymbol{G},\ell} \right) + \mathrm{F}_o \tag{11}$$

Where $\mathrm{F}''_{\text{attn}}$ and $\mathrm{F}''_{\text{ffn}}$ are the corresponding FLOPs scales after selecting the largest value in $\{\boldsymbol{\alpha}^a_\ell, \boldsymbol{\alpha}^m_\ell, \boldsymbol{\alpha}^h_\ell, \boldsymbol{\alpha}^n_\ell\}$ in Eq. 10 respectively. For both stages, dynamic compression is trained with group-level gates, and the γ is set to 100.

Methods	Efficiency		Pruning Options		Top-1 ACC.(%)↑
	Params.↓	FLOPs↓	Static	Dynamic	
ViT-base [2]	86M	17.6G	-	-	77.9%
DeiT-S [18]	22M	4.6G	-	-	79.8%
LeViT-256 [54]	18.9M	1.1G	-	-	80.1%
T2T-14 [5]	21.5M	4.8G	-	-	81.5%
DynamicViT (DeiT-S) [11]	22M	3.4G	-	✓	78.3%
A-ViT (DeiT-S) [2]	22M	3.6G	-	✓	78.6%
MIA-Former (DeiT-S) [14]	22M	4G	-	✓	78.6%
Evo (LeViT-256) [12]	19M	-	-	✓	78.8%
USDC (DeiT-S)	**19M**	**3.35G**	✓	✓	**79.0%**
USDC (T2T-14)	**19.9M**	**3.4G**	✓	✓	**80.6%**
AdaViT (T2T-19) [13]	32M	3.9G	-	✓	81.1%
USDC (T2T-14)	**20.3M**	**3.7G**	✓	✓	**81.2%**

Table 1: Our USDC outperforms these recent baseline pruning methods by a clear margin, achieving a better balance between the model efficiency and the compression ratios.

4. Experiments

Datasets and Benchmarks. We conduct experiments for image classification tasks on ImageNet-1K [19] datasets. We use DeiT [18] and T2T-VIT [5] as the backbone models due to their strong performance on the image classification task, which both are vision transformer-based models. We compare the Top-1 accuracy and the inference floating point operations (FLOPs) on ImageNet-1K validation dataset.

Main Results We compare USDC with several other dynamic compression methods that achieve state-of-the-art results on vision transformer models. DynamicViT [11] prune redundant tokens via halting decisions relaxed by Gumbel-softmax. A-ViT early learns halting via adaptive computation time [55] to stop tokens early. AdaViT [13] uses three linear layers as the dynamic gate network to control the token, head, and block separately. MIA-Former applies two-layer CNN with one-layer linear as MIA-Controller to dynamic skipping on the head, depth, and token of ViT models. All of those compression methods are dynamic pruning types. Most of the previous dynamic methods on ViT models need to apply additional parameters for the dynamic decision network, which increases the overall number of parameters than the original ViT models [11, 13, 14].

The comparison with previous work is summarized in Tab. 1. To the best of our knowledge, USDC is the first to unify the static and dynamic compression together, the reduction of parameters under static dynamic compression can offset the added parameters on the dynamic gates network. From Tab. 1, we can notice that all USDC models can reduce the overall parameter compared with backbone models and the other dynamic network method. In comparison, USDC can cut down parameters and FLOPs while achieving the best performance of **79.0%** on DeiT-small compared with DynamicViT, A-ViT, and MIA-Former. **More experiments and ablations are in the appendix.**

5. Conclusions

In this paper, we propose a unified static and dynamic compression framework USDC for the vision transformer, which can be jointly optimized in an end-to-end manner. In the USDC framework, the gate network for each encoder layer will be automatically selected by the NAS for a better trade-off between performance and efficiency. We also proposed the group-level gate augmentation strategy and introduced it into the training for the performance consistency between training and inference. The comparisons with the state-of-the-art compression methods show the effectiveness of USDC. Additionally, the ablation studies show that the group-level strategy and gate network search can benefit the performance of compression. However, our training procedures are still complicated, in the future, we will further explore more efficient and effective unified dynamic and static compression framework designs.

Effects of G	Batch	Top-1 Accuracy (%)↑		
		Manual	Manual	Search
USDC (DeiT-S)	256	78.44	78.32	**78.89**
	32	78.44	78.30	**78.93**
	2	78.46	78.30	**78.98**
FLOPs of G_*	-	0.89M	**0.02M**	0.75M
FLOPs of model	-	3.37G	3.4G	**3.35G**

Table 2: **Effects of different dynamic decision networks.** Batch means the inference batch size during evaluation. The first column manually chooses $G_{:,0}$ as a dynamic decision network for all encoder layers. And the second column manually chooses $G_{:,2}$ for all layers. The third column automatically searches dynamic decision networks as described at Seq. 3.2.

Pruning Options	Static	Dynamic	Static & Dynamic
Top-1 ACC.	77.97%	78.13%	**78.96%**
FLOPs	3.40G	3.36G	**3.35G**

Table 3: Comparisons of different pruning options on DeiT-S demonstrate the superiority of our joint optimization.

Comparisons of different pruning options. In Tab. 3, we regard USDC as a combined static and dynamic compression method, and we compare USDC with only static compression methods and only dynamic compression methods on the DeiT-S model. It can be noticed that the joint static and dynamic method in USDC has achieved the highest effect. A similar illustration is also shown in Fig. 5.

C. More Compression Comparisons

To further demonstrate the superiority of our method when the compression rate is smaller than 50%, we conduct the USDC on the T2T-ViT-19 [5] model. From Tab. 4, we can find that USDC reduces the 55.3% FLOPs of the original baseline T2T-ViT-19 while the drop of Top-1 accuracy on Imagenet-1K [19] dataset is only 0.6%. Our USDC framework outperforms the previous dynamic compression method AdaViT [13] with fewer parameters and FLOPs. It should be noticed that AdaViT takes three linear layers as the dynamic decision network for each transformer layer in T2T-ViT-19, thus the number of parameters in AdaViT is larger than the original T2T-ViT-19.

Table 4: The comparisons on the T2T-ViT-19 model.

Method	Params.	FLOPs	Top-1 Acc.
Baseline (T2T-ViT-19) [5]	39.2M	8.5G	81.9%
Ada-ViT (T2T-ViT-19) [13]	>39.2M	3.9G	81.1%
USDC (T2T-ViT-19)	**34.4M**	**3.8G**	**81.3%**

D. Ablations for sub-groups split methods

Whether the samples within a mini-batch ought to use the same gate controlling at any location of ViT during the training of the dynamic network is a heuristic question. Different from **sample-level** strategy [2, 13, 27] and **batch-level** strategy, we propose a **group-level** strategy to augment dynamic gates. Group-level split batch by sub-groups and calculate gate under the sub-group, which can simulate various smaller batch sizes and mitigate the performance deviation caused by the inconsistency of batch size between train and inference stages. USDC recursively split the gates features into the different groups by a logarithm of 2.

For **group-level** strategy, we trained the USDC method by splitting sub-groups on average, on random, and our recursive split method separately. The training is on the DeiT-small model on the Imagenet-1K dataset, the training batch size is 256 and all other parameter settings are the same. As shown in Tab. 5, we compare different sub-groups split method, the The top-1 accuracy on Imagenet-1K of our recursive split method separately is better than the average and random methods.

Table 5: The comparisons of different sub-groups split methods for group-level gate augmentation strategy. The first column splits sub-groups uniformly with step size 32. The second column splits sub-groups uniformly with step size 8. The third column splits the sub-groups randomly with step size ranges in $[1, 64]$. The fourth column (Ours) splits sub-groups recursively by a logarithm of 2.

Model	B	**Top-1 Accuracy (%)**			
		Avg-32	Avg-8	Random	**Ours**
USDC (DeiT-S)	256	77.13	77.43	77.86	**78.89**
	64	77.11	77.62	77.44	**78.90**
	32	77.10	77.62	77.45	**78.93**
	8	77.05	77.65	77.50	**78.96**
	2	77.00	77.62	77.49	**78.98**
	1	76.91	77.60	77.49	**78.96**
FLOPs	-	3.30G	3.36G	3.30G	3.35G

E. Visualizations

We illustrate the structures of the compressed DeiT-Small model by USDC at Fig. 5. We trained the model in Fig. 5 by unified static and dynamic compression described in the main text. We can notice that the head number of MHSA and the hidden dimension of FFN were reduced by static compression, and some blocks were pruned by static compression of USDC. Meanwhile, the dynamic compression of USDC skipped each block adaptively according to the input features of each transformer layer. The FLOPs of all 12 dynamic decision networks together are only 0.45M, and the FLOPs of the original DeiT-small is 4.6G. As shown in Fig. 5, the remaining FLOPs achieved by only the static compression part is 74.9%. The final remaining FLOPs achieved by joint static and dynamic compression is 64.8%.

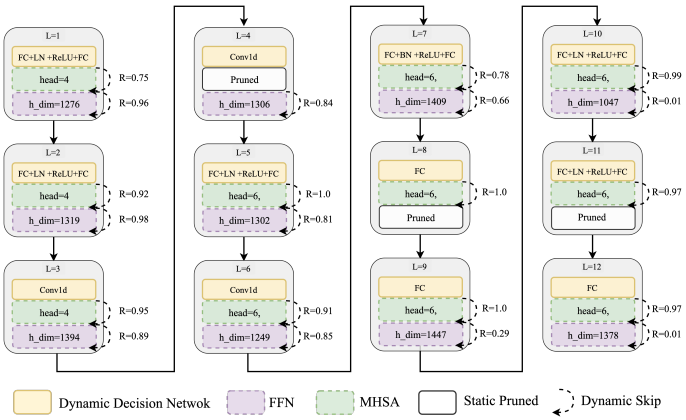

Figure 5: An architecture illustration for the compressed DeiT-small model using our USDC. The L is the index of transformer encoder layers. The $head$ is the head number of the MHSA block. The h_dim is the hidden dimension of FFN blocks. The R is the dynamically executing rate on average for each block. The remaining FLOPs achieved by static compression is 74.9%. The final remaining FLOPs achieved by joint static and dynamic compression is 64.8%.