SLAB: Efficient Transformers with Simplified Linear Attention and Progressive Re-parameterized Batch Normalization

Jialong Guo [1 *] **Xinghao Chen** [1 *] **Yehui Tang** [1] **Yunhe Wang** [1]

Abstract

Transformers have become foundational architectures for both natural language and computer vision tasks. However, the high computational cost makes it quite challenging to deploy on resource-constraint devices. This paper investigates the computational bottleneck modules of efficient transformer, *i.e.*, normalization layers and attention modules. LayerNorm is commonly used in transformer architectures but is not computational friendly due to statistic calculation during inference. However, replacing LayerNorm with more efficient BatchNorm in transformer often leads to inferior performance and collapse in training. To address this problem, we propose a novel method named PRepBN to progressively replace LayerNorm with re-parameterized BatchNorm in training. Moreover, we propose a simplified linear attention (SLA) module that is simple yet effective to achieve strong performance. Extensive experiments on image classification as well as object detection demonstrate the effectiveness of our proposed method. For example, our SLAB-Swin obtains 83.6% top-1 accuracy on ImageNet-1K with 16.2ms latency, which is 2.4ms less than that of Flatten-Swin with 0.1% higher accuracy. We also evaluated our method for language modeling task and obtain comparable performance and lower latency. Codes are publicly available at https://github.com/xinghaochen/SLAB and https://github.com/mindspore-lab/models/.

1. Introduction

Introduced initially for tasks in natural language processing (Vaswani et al., 2017), transformer architecture has rapidly emerged as a preeminent model in the landscape of language models. Its influence has significantly expanded with the introduction of Vision Transformer (ViT) (Dosovitskiy et al., 2020), illustrating the efficacy and versatility of transformer-based architectures. These architectures have demonstrated their capability to achieve competitive performance benchmarks in comparison to convolutional neural networks (CNNs) across diverse vision tasks (Han et al., 2022; Wang et al., 2022; Zheng et al., 2023; Tang et al., 2023a; Carion et al., 2020; Xu et al., 2023). Due to its powerful performance, transformer has become the mainstream architecture in deep learning. However, the computational demands of transformer architecture pose a significant challenge, which is predominantly due to the quadratic computational complexity of its attention mechanism and the necessity for online statistic computation of LayerNorm component.

Numerous efforts have been directed towards enhancing the efficiency of transformer architecture (Tang et al., 2024; Wu et al., 2023; Tang et al., 2023b). Several approaches have sought to mitigate computational complexity by limiting the scope of token interactions within self-attention mechanisms, such as downsampling the key and value matrices (Wang et al., 2021), implementing sparse global attention patterns (Child et al., 2019), and computing self-attention within smaller windows (Tu et al., 2022; Liu et al., 2021; Dong et al., 2022). Meanwhile, linear attention emerges as an alternative strategy to enhance computational efficiency by breaking down the attention mechanism into linear computational cost (Cai et al., 2022; Han et al., 2023), yet it is still a challenging task to obtain a good balance between efficiency and accuracy. Moreover, there are some explorations into substituting LayerNorm (LN) with BatchNorm (BN) within transformers, motivated by the additional computational overhead LayerNorm incurs during inference. Yang *et al*. (2022) propose to add a BatchNorm layer in-between the two linear layers in the feed forward network to stabilize the training. However, there still exists a performance gap between the LayerNorm-based and BatchNorm-based transformers.

In this paper, we focus on obtaining efficient transformer architectures by digging deep into the computational in-

[*]Equal contribution [1]Huawei Noah's Ark Lab. Correspondence to: Xinghao Chen <xinghao.chen@huawei.com>, Yunhe Wang <yunhe.wang@huawei.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

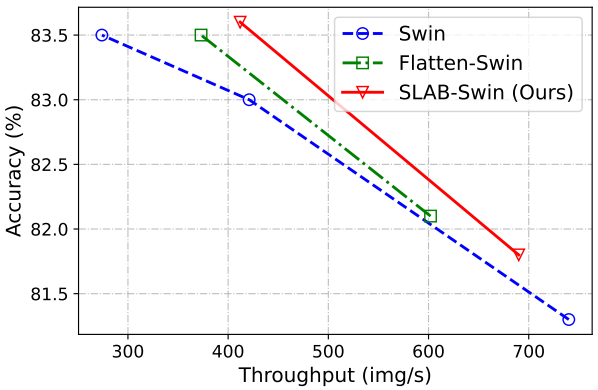


Figure 1. Comparisons of different methods on ImageNet.

efficient modules, *i.e.*, normalization layers and attention modules. We first explore to replace LayerNorm with BatchNorm to accelerate inference for transformer. BatchNorm leads to lower inference latency but may cause training collapse and inferior performance, while LayerNorm could stabilize the training yet has extra computational cost during inference. To this end, we first propose a progressive strategy to gradually replace LayerNorm with BatchNorm by using a hyper-parameter to control the proportion of both normalization layers. Initially the transformer architecture is dominated by the LayerNorm and gradually transits to pure BatchNorm at the end of training. This strategy effectively mitigates the risk of training collapse and also eliminating the need for calculating statistics during inference. In addition to the progressive strategy, we also propose a novel re-parameterization formula for BatchNorm (RepBN), to enhance training stability and overall performance.

Furthermore, the computational cost of attention is critical for efficient transformer and prior methods struggle to obtain good balance of efficiency and accuracy. To this end, we propose a simplified linear attention (SLA) module which utilizes ReLU as the kernel function and incorporate a depthwise convolution to perform local feature enhancement. The proposed attention mechanism is more efficient than prior linear attention but still attains comparable performance.

We extensively evaluate our proposed method for various architectures on various benchmarks. Our progressive re-parameterized BatchNorm shows strong performance for image classification and object detection tasks, obtaining similar accuracy with lower inference latency. Moreover, coupled with the progressive RepBN and simplified linear attention module, our SLAB transformer achieves competitive accuracy compared to Flatten transformer with improved computational efficiency. For example, SLAB-Swin-S achieves 83.6% Top-1 accuracy on ImageNet-1K with 16.2ms latency, which is 2.4ms less than that of Flatten-Swin-S with 0.1% higher accuracy. We also evaluated our method for language modeling task and obtain comparable performance and lower inference latency.

2. Related Work

2.1. Efficient Architecture for Transformers

With the advent of the pioneering Vision Transformer (ViT) (Dosovitskiy et al., 2020), the potential of the transformer architecture for computer vision tasks has been greatly explored. Various researchers are devoted to this field to make transformer-based architecture more efficient and powerful. Touvron *et al*. (2021) propose DeiT which utilizes distillation to achieve strong performance with training only on ImageNet1K. Liu *et al*. (Liu et al., 2021) propose Swin Transformer, which introduces shifted windowing scheme and brings greater efficiency. As the self-attention computation is limited to a small window, this transformer has linear computational complexity. Several works improve the design of sparse pattern to enhance the interaction of each token, such as CSwin (Dong et al., 2022). Besides, the dynamic attention mechanism tries to control the key/value interact with query adaptive to data, such as DAT++ (Xia et al., 2023) and BiFormer (Zhu et al., 2023).

Apart from the above methods, linear attention is a popular research direction to reduce the computational complexity for transformer. Many effective mechanisms have been proposed to replace the softmax function. For example, Performers (Choromanski et al., 2020) uses positive orthogonal random features approach to approximate softmax. Hydra attention (Bolya et al., 2022) selects the cosine similarity as kernel function. Flatten Transformer (Han et al., 2023) designs a focused function to improve the focus ability of linear attention.

2.2. Normalization for Transformers

Normalization is known as an useful method to make training stable and boost performance. Nowadays, a variety of normalization methods have been proposed, such as BatchNorm (Ioffe & Szegedy, 2015), LayerNorm (Ba et al., 2016), InstanceNorm (Ulyanov et al., 2016), GroupNorm (Wu & He, 2018), MABN (Yan et al., 2020) and UN (Yang et al., 2022). BatchNorm is widely used in convolutional networks and LayerNorm is commonly utilized for networks such as transformer and LSTM.

Normalization could be categorized into offline methods and online methods according to whether the mean and variance need to be computed at inference time (Yang et al., 2022). Online methods are usually batch-irrelevant like LayerNorm, InstanceNorm and GroupNorm. These methods compute the statistics in both training and inference. LayerNorm is a commonly used in transformer architecture.

Offline methods are batch-related like BatchNorm and UN, in which the batch dimension is concluded in the calculations of both mean and variance (Yao et al., 2021). As the mean and variance are pre-computed in inference, offline

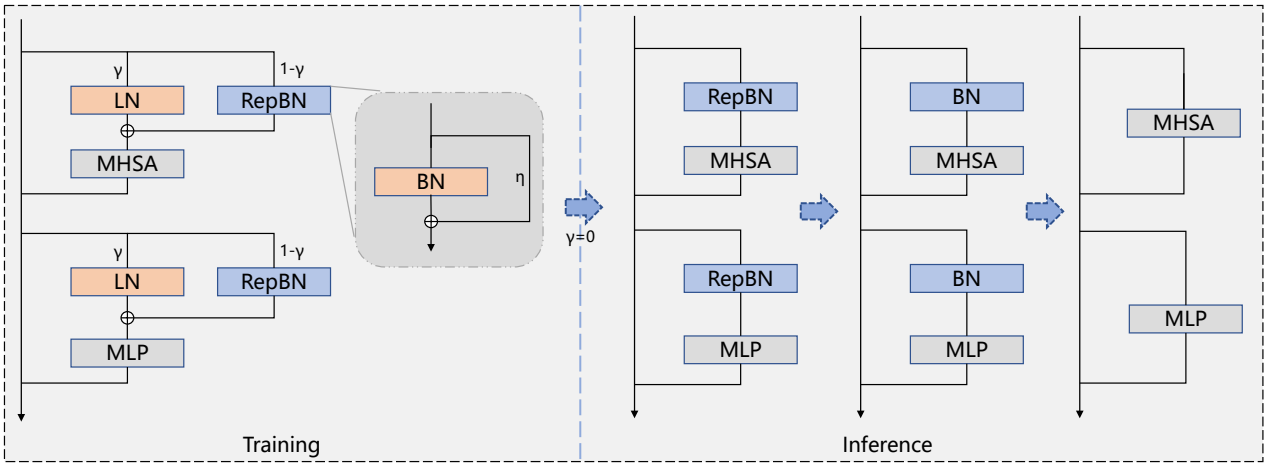


Figure 2. The overall framework of our proposed Progressive Re-parameterized BatchNorm. (a) During training, we progressively replace LayerNorm with RepBN, which is a new re-parameterization formula of BatchNorm to further improve the performance. (b) We could get $\gamma = 0$ during inference, thus the transformer block transits to a RepBN-based architecture, which could further be re-parameterized to BatchNorm and merged with linear layers.

normalization can be fused into adjacent linear operations. During inference, there will be no offline normalization operations and the inference time will be reduced. However, offline methods usually face the problem of performance degradation and training collapse while using in transformer. To address this problem, Yao *et al*. (2021) proposes to add a BatchNorm layer in-between the two linear layers in the MLP block that makes training statistics stable. Yang *et al*. (2022) finds that the issue is caused by abnormal behaviors of activation statistics, and proposes a tailored fluctuation smoothing strategy and an adaptive outlier filtration strategy to boost performance and stable training.

3. Preliminaries

Given the input N tokens $X \in \mathbb{R}^{N\times C}$, where C is the feature dimension, the general architecture of transformer block can be written as:

$$\begin{aligned} X &= X + \mathrm{Attn}(\mathrm{Norm}(X)), \\ X &= X + \mathrm{MLP}(\mathrm{Norm}(X)), \end{aligned} \tag{1}$$

where $\mathrm{Attn}(\cdot)$ calculates the attention scores, $\mathrm{MLP}(\cdot)$ denotes multilayer perceptron and $\mathrm{Norm}(\cdot)$ is the normalization function. In the default configuration of transformer block, $\mathrm{Norm}(\cdot)$ is usually a LayerNorm operation and $\mathrm{Attn}(\cdot)$ is the softmax-based attention mechanism (Vaswani et al., 2017).

Attention plays an important role in Transformer. Denote query, key and value matrix as $Q, K, V \in \mathbb{R}^{N\times C}$, softmax attention computes the pairwise similarity between queries and keys firstly, and leads to the quadratic computation complexity $O(N^2C)$ in relation to the number of queries and keys N. This makes transformer computationally expensive especially in dealing with tasks that have a long sequence input. Linear attention aims to decouple the softmax function with proper approximation or instead it with other kernel function to compute K^TV first. With this change in computation order, the computation complexity becomes $O(NC^2)$, which is linearly related to the number of queries and keys N.

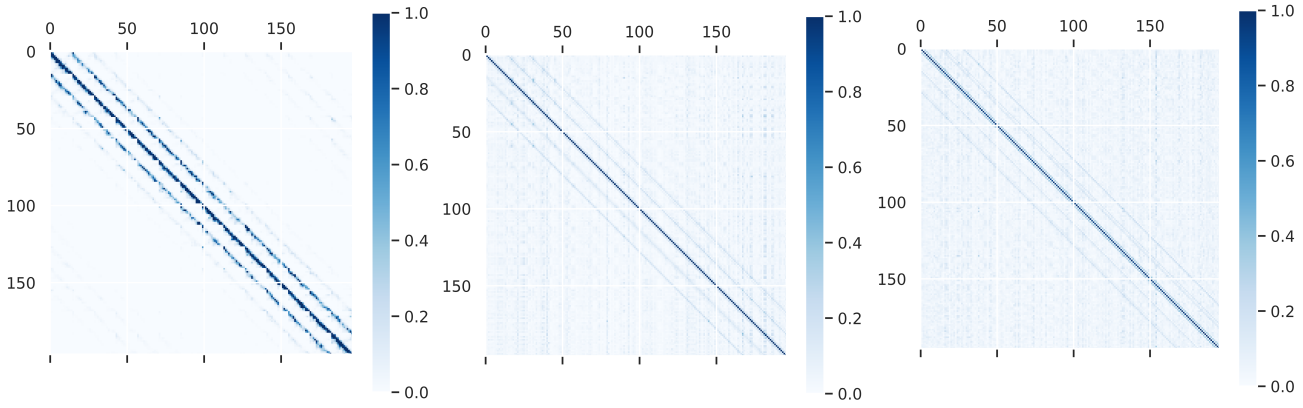
However, LayerNorm occupies unnegligible portion of latency since it requires statistic calculation during inference. Therefore, in this paper we explore to leverage BatchNorm for building efficient transformers, which only exists in training and could be merged with preceding or sequential linear layers. Moreover, the attention module plays the most important part for transformers and the softmax-based attention mechanism is computational inefficient due to its quadratic computation complexity. In this paper, we propose a simple yet efficient form of attention, which greatly reduce the latency but also remains strong performance on various vision tasks.

4. Methods

In this paper, we focus on building efficient transformers and propose a series of strategies, including a progressive strategy to replace the LayerNorm (LN) with the re-parameterized BatchNorm (BN) and the simplified linear attention (SLA) module. The proposed SLAB transformers obtains strong performance compared with prior methods while enjoying more computational efficacy.

4.1. Progressive Re-parameterized BatchNorm

LayerNorm requires statistic calculations in both training and inference, thus significantly hinders the running speed of transformers. On contrast, BatchNorm could be sim-



(a) DeiT (b) Flatten Transformer (c) SLAB (Ours)

Figure 3. Attention map (196×196) from the 4rd block of the model based on DeiT-T. (a) Attention map of DeiT-T is full-rank. (b) With the help of depth-wise convolution, linear attention in Flatten Transformer has a high rank. (c) As simplified linear attention and progressive re-parameterized BatchNorm are applied in transformer, the model still keeps a high rank.

ply merged with linear layers during inference and is more suitable for efficient architectures. However, directly leveraging BatchNorm for transformers brings unsatisfactory performance (Yao et al., 2021). To this end, we propose to progressively replace LayerNorm with BatchNorm during training, and also propose a new re-parameterization formula of BatchNorm to further improve the performance, as shown in Figure 2.

Re-parameterized BatchNorm. The proposed RepBN is formulated as:

$$\mathrm{RepBN}(X) = \mathrm{BN}(X) + \eta X, \tag{2}$$

where η is a learnable parameter that is jointly trained in an end-to-end manner. Once the training is done, the RepBN could be re-parameterized as a norm form of BN, as shown in Lemma 4.1.

Lemma 4.1. *Denote a BN layer with mean μ, standard deviation σ, rescale and shift parameters α and β as $\mathrm{BN}(X; \mu, \sigma, \alpha, \beta)$. We can re-parameterize the RepBN in Eq. 2 as:*

$$\mathrm{RepBN}(X; \mu, \sigma, \alpha, \beta) = \mathrm{BN}(X; \mu, \sigma, \alpha + \eta\sigma, \beta + \eta\mu). \tag{3}$$

Proof.

$$\begin{aligned}
&\mathrm{RepBN}(X; \mu, \sigma, \alpha, \beta) = \mathrm{BN}(X; \mu, \sigma, \alpha, \beta) + \eta X \\
&= \frac{X-\mu}{\sigma}\alpha + \beta + \eta X = \frac{X-\mu}{\sigma}\alpha + \beta + \frac{X}{\sigma}\sigma\eta \\
&= \frac{X-\mu}{\sigma}\alpha + \beta + \frac{X-\mu}{\sigma}\sigma\eta + \mu\eta \\
&= \frac{X-\mu}{\sigma}(\alpha + \eta\sigma) + (\beta + \eta\mu) \\
&= \mathrm{BN}(X; \mu, \sigma, \alpha + \eta\sigma, \beta + \eta\mu).
\end{aligned} \tag{4}$$

□

Based on Lemma 4.1, the distribution of RepBN's output is control by $\alpha + \eta\sigma$ and $\beta + \eta\mu$, which is corresponds to the variance and mean. RepBN can recover the distribution with the help of σ and μ.

Meanwhile, when $\alpha = 0, \beta = 0$, it is equivalent to BatchNorm being skipped. When $\eta = 0$, RepBN is converted into pure BatchNorm.

Progressive LN $\rightarrow$ RepBN. To facilitate the training of a pure BN-based transformers, we propose to progressively transit the LN to RepBN during training, *i.e.*,

$$\mathrm{PRepBN}(X) = \gamma\mathrm{LN}(X) + (1-\gamma)\mathrm{RepBN}(X), \tag{5}$$

where γ is a hyper-parameter to control the output of different normalization layers. Generally $\gamma = 1$ at the begin of training when the LN dominates the architecture, and $\gamma = 0$ at the end of training to make sure it transits to a pure BN-based transformer. We utilize a simple yet effective decay strategy for γ:

$$\gamma = \frac{T - T_{cur}}{T}, \gamma \in [0, 1], \tag{6}$$

where T is the total steps of training with LayerNorm and T_{cur} is the current step. This progressive strategy eases the difficulty of training a pure BN-based transformer and thus leads to strong performance on various tasks.

There are some other decay strategies for attenuating the value of γ gradually, such as cosine decay and step decay. Empirically, we find that the linear strategy is one of the more effective and simpler.

4.2. Simplified Linear Attention

Attention module is the most import part in a transformer network, which is generally formulated as:

$$Q = XW_Q, K = XW_K, V = XW_V,$$
$$O_i = \sum_{j=1}^{N} \frac{\mathrm{Sim}(Q_i, K_j)}{\sum_j \mathrm{Sim}(Q_i, K_j)} V_j, \tag{7}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{C\times C}$ project the input tokens to query, key and value tensors, respectively. $\mathrm{Sim}(\cdot,\cdot)$ denotes the similarity function. For the original form of attention, the similarity function is:

$$\mathrm{Sim}_{\mathrm{softmax}}(Q_i, K_j) = \exp(\frac{Q_i K_j^T}{\sqrt{C}}), \tag{8}$$

this softmax-based attention leads to high computational complexity. Several recent methods investigate the usage of linear attention to remove the softmax calculation thus improve the efficacy of transformers (Han et al., 2023). However, these methods still suffer quite complex design and are not computation efficient enough. In this paper, we propose a simplified linear attention (SLA) which is formulated as follow:

$$\mathrm{Sim}_{SLA}(Q_i, K_j) = \mathrm{ReLU}(Q_i)\,\mathrm{ReLU}(K_j)^T,$$
$$\tilde{\mathrm{O}}_i = \sum_{j=1}^{N} \frac{\mathrm{Sim}_{SLA}(Q_i, K_j)}{\sum_j \mathrm{Sim}_{SLA}(Q_i, K_j)} V_j, \tag{9}$$
$$\mathrm{O}_{SLA} = \tilde{\mathrm{O}} + \mathrm{DWC}(V),$$

where $DWC(\cdot)$ denotes a depth-wise convolution. It is a simple yet efficient linear attention since it also enjoys the decoupling computation order by computing $K^T V$ first, and leads to great complexity reduction. Moreover, only ReLU function and depth-wise convolution are explored and both operations are computation friendly in most hardware.

To demonstrate that our method still maintains feature diversity, we visualize the effect of attention map on DeiT-T that applied the strategy of progressive re-parameterized BatchNorm and simplified linear attention (SLAB), as shown in Figure 3. It can be find that a high rank is still kept for our proposed method, demonstrating its good capacity for capturing attention information.

5. Experiments

In this section, we evaluate our method on various computer vision tasks including image classification, object detection and instance segmentation and also on language modeling task. We conduct extensive experiments for various backbones with our proposed progressive re-parameterized BatchNorm and simplified linear attention module. We train our model on ImageNet-1K (Deng et al., 2009) for image classification, and evaluate the effect of object detection and instance segmentation tasks on COCO dataset (Lin et al., 2014). At last, we ablate the important design elements of our proposed method on classification task.

5.1. Image Classification

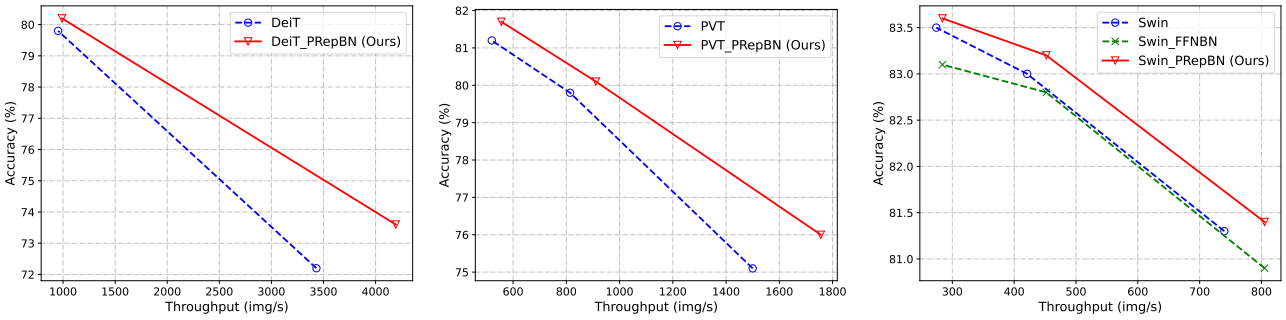
Settings. For image classification, we adhere to the configuration outlined in (Touvron et al., 2021). We train all models for 300 epochs with AdamW optimizer, incorporating a cosine decay learning rate scheduler with 20 epochs of linear warm-up. The batch size is set to 1024, the initial learning rate is 0.001, and the weight decay value is 0.05. In the case of models utilizing the proposed progressive re-parameterized BatchNorm, a reduced droppath (Larsson et al., 2016) rate is applied. The linear decay steps T for PRepBN slightly varies across different backbones. Due to the variance shift induced by droppath, we freeze the model parameters and exclusively update the statistics of re-parameterized BatchNorm for 10 epochs at the end of training. We also demonstrate the effectiveness of our method with both progressive re-parameterized BatchNorm and simplified linear attention. We follow the setting of (Han et al., 2023) on macro architecture design and training. All reported results of throughput/latency are obtained on a single V100 GPU. For classification task, we measure FLOPs as well as the throughput/latency for the image resolution of 224×224.

Results on image classification task. Table 1 presents the results of different backbones with our PRepBN normalization. Our proposed PRepBN demonstrates comparable or even superior performance when compared with LayerNorm. More specifically, the models using our PRepBN as normalization layer exhibit performance improvements ranging from 0.1% to 1.4%. Notably, PRepBN is amenable to fusion with other linear operations, allowing it to obtain more efficient inference. We further compare our method with BN+FFNBN (Yao et al., 2021) which also aims to train transformer model with BatchNorm. It can be seen that our PRepBN achieves consistent improvements on different backbones. For example, our proposed PRepBN achieves 80.2% top-1 accuracy on DeiT-S model, which is 1.4% better than the BN+FFNBN method. For swin transformer, our PRepBN brings +0.5%, +0.4% and +0.5% accuracy gain than BN+FFNBN on Swin-T, Swin-S and Swin-B models. As shown in Figure 4, we present a comparative analysis of our method across DeiT-based, PVT-based, and Swin-based models. It is evident that transformers equipped with our PRepBN achieve higher throughput while maintaining similar accuracy levels.

Table 2 presents the performance of our SLAB transformer, which is powered by our proposed progressive re-parameterized BatchNorm and simplified linear attention

Table 1. Comparison of different normalizations for various transformer architectures on ImageNet1K.

Method	Normalization	FLOPs (G)	Throughput	Top-1 Acc. (%)
DeiT-T (Touvron et al., 2021)	LN (Default)	1.3	3432	72.2
	PRepBN (Ours)	1.3	**4194**	**73.6**
DeiT-S (Touvron et al., 2021)	LN (Default)	4.6	952	79.8
	BN+FFNBN (Yao et al., 2021)	4.6	990	78.8
	PRepBN (Ours)	4.6	**990**	**80.2**
PVT-T (Wang et al., 2021)	LN (Default)	1.9	1500	75.1
	PRepBN (Ours)	1.9	**1756**	**76.0**
PVT-S (Wang et al., 2021)	LN (Default)	3.8	814	79.8
	PRepBN (Ours)	3.8	**911**	**80.1**
PVT-M (Wang et al., 2021)	LN (Default)	6.7	520	81.2
	PRepBN (Ours)	6.7	**556**	**81.7**
Swin-T (Liu et al., 2021)	LN (Default)	4.5	740	81.3
	BN+FFNBN (Yao et al., 2021)	4.5	805	80.9
	PRepBN (Ours)	4.5	**805**	**81.4**
Swin-S (Liu et al., 2021)	LN (Default)	8.7	421	83.0
	BN+FFNBN (Yao et al., 2021)	8.7	452	82.8
	PRepBN (Ours)	8.7	**452**	**83.2**
Swin-B (Liu et al., 2021)	LN (Default)	15.4	274	83.5
	BN+FFNBN (Yao et al., 2021)	15.4	284	83.1
	PRepBN (Ours)	15.4	**284**	**83.6**



(a) DeiT (b) PVT (c) Swin

Figure 4. Comparisons of accuracy and throughput for different methods on ImageNet1k.

module. We compare our model with Flatten transformer, which utilizes focused linear attention for higher efficiency. Experiments on various architectures including DeiT (Touvron et al., 2021), PVT (Wang et al., 2021), CSwin (Dong et al., 2022) and Swin (Liu et al., 2021) demonstrate than our SLAB transformer obtains better performance than Flatten transformer. More specifically, our SLAB-Swin-T model obtains 83.6% top-1 accuracy on ImageNet-1K with 16.2ms latency, which is 2.4ms less than that of Flatten-Swin with 0.1% higher accuracy. Our models are more computational efficient mainly due to more hardware friendly normalization layers as well as the simplified linear attention module. Figure 1 also shows the trade-off between accuracy and latency of our SLAB transformer, Flatten transformer (Han et al., 2023) and the original Swin transformer, which demonstrate better performance of our model.

5.2. Object Detection

Settings. We use Mask R-CNN (He et al., 2017) to evaluate the effectiveness of our method on COCO dataset for object detection and instance segmentation tasks. The backbones used in Mask R-CNN are pretrained on ImageNet-1K. All models are trained for $1\times$ schedule, *i.e.*, 12 epochs. The latency is measured with a batch size of 1 on V100 GPU for a average value of 100 rounds.

Results on object detection task. We compare our proposed PRepBN with standard LayerNorm for various backbones including Swin and PVT for object detection task. The results are shown in Table 3. It reveals that our method achieves quite comparable performance with original models equipped with LayerNorm. Taking advantages of offline normalization, the models with our proposed PRepBN ob-

Table 2. Comparison of different linear transformers on ImageNet1K.

Method	FLOPs (G)	Latency (ms)	Top-1 Acc. (%)
Flatten-DeiT-T (Han et al., 2023)	1.1	15.2	74.1%
SLAB-DeiT-T (Ours)	1.1	**9.6**	**74.3%**
Flatten-DeiT-S (Han et al., 2023)	4.4	15.5	80.4%
SLAB-DeiT-S (Ours)	4.4	**10.4**	**80.0%**
Flatten-PVT-T (Han et al., 2023)	2.0	10.8	77.8%
SLAB-PVT-T (Ours)	2.0	**8.0**	**76.5%**
Flatten-CSwin-T (Han et al., 2023)	4.3	32.4	83.1%
SLAB-CSwin-T (Ours)	4.3	**29.3**	**82.8%**
Flatten-Swin-T (Han et al., 2023)	4.5	10.9	82.1%
SLAB-Swin-T (Ours)	4.5	**8.7**	**81.8%**
Flatten-Swin-S (Han et al., 2023)	8.8	18.6	83.5%
SLAB-Swin-S (Ours)	8.7	**16.2**	**83.6%**

Table 3. Mask R-CNN Object Detection & Instance Segmentation on COCO.

Method	Schd.	Lat. (ms)	$\mathbf{AP^b}$	AP^b_{50}	AP^b_{75}	AP^b_s	AP^b_m	AP^b_l	$\mathbf{AP^m}$	AP^m_{50}	AP^m_{75}	AP^m_s	AP^m_m	AP^m_l
Swin-T (Liu et al., 2021)	1×	51.0	43.1	66.0	47.0	27.3	46.4	56.0	39.4	62.8	42.0	23.0	42.9	53.7
Swin-T-PRepBN (Ours)	1×	**43.0**	**42.9**	65.8	46.8	27.7	46.2	55.4	**39.3**	62.6	41.9	23.1	42.8	53.6
Swin-S (Liu et al., 2021)	1×	63.1	46.2	68.7	50.9	30.2	49.7	60.8	41.7	65.6	44.5	25.1	45.5	57.3
Swin-S-PRepBN (Ours)	1×	**57.3**	**45.9**	68.4	50.2	29.8	49.3	60.2	**41.4**	65.0	44.7	24.9	44.9	57.0
PVT-T (Wang et al., 2021)	1×	46.0	36.6	58.9	39.2	21.3	38.9	48.9	34.9	56.2	37.0	19.6	37.2	48.2
PVT-T-PRepBN (Ours)	1×	**43.2**	**36.5**	59.0	39.2	20.9	38.4	50.1	**34.4**	55.7	36.5	18.7	36.3	48.9
PVT-S (Wang et al., 2021)	1×	64.0	40.5	63.2	44.1	23.4	43.4	54.9	37.9	60.2	40.6	20.8	40.6	52.6
PVT-S-PRepBN (Ours)	1×	**59.6**	**40.6**	63.3	43.9	24.4	43.1	55.4	**38.0**	60.6	40.5	21.4	40.5	53.5
Flatten-Swin-T (Han et al., 2023)	1×	60.0	44.2	67.3	48.5	29.4	47.5	57.0	40.2	63.8	43.0	24.5	43.8	54.7
SLAB-Swin-T (Ours)	1×	**54.1**	**43.9**	66.5	48.1	28.6	47.4	56.7	**40.1**	63.3	43.1	24.3	43.8	54.2

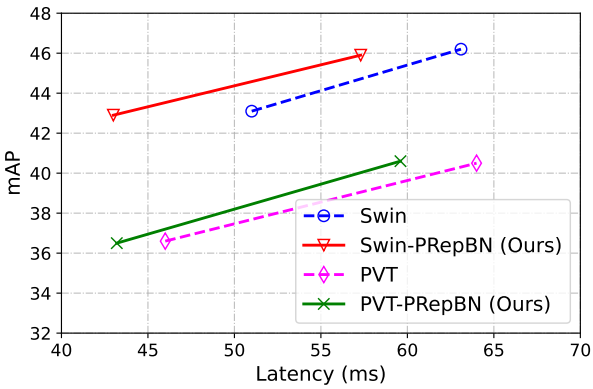


Figure 5. Comparison of different normalization on COCO.

tain lower inference latency. For example, the latency of Mask R-CNN exhibits a reduction from 64ms to 59.6ms when PVT-S backbone is equipped with our PRepBN and the accuracy for object detection and instance segmentation is similar. A more clear visualization for the trade-off between mAP and latency is also presented in Figure 5, which demonstrates that our proposed method achieves better overall performance on object detection.

5.3. language modeling

Settings. We also evaluate our proposed method on language modeling task based on Adaptive Inputs (Baevski & Auli, 2018). We train our models on the Wikitext-103 (Merity et al., 2016) dataset, which contains over 100 million tokens. We set the number of tokens per GPU to 4096 and train on 8 GPUs. The number of tokens per sample is limit to 512. We also apply our method on LLaMA-350M model, following the similar architecture and training settings as prior work (Yang et al., 2023; He et al., 2024).

Results on language modeling task. As shown in Table 4, our PRepBN achieves similar perplexity with the model equipped with LayerNorm, while the latency reduces from 13.9 ms to 12.9 ms per token. Besides, We apply our PRepBN on more modern large language models such as LLaMA, which adopts the variant of LayerNorm that removes the computation of mean values, *i.e.*, RMSNorm. As shown in Table 5, our method successfully boost the throughput from 44.0 to 50.4 tokens per second on V100 GPU, and obtains even slightly better average accuracy. These results demonstrate the effectiveness of our proposed PRepBN on language modeling task.

Table 4. Results of perplexity on Wikitext-103 dataset. 256/480 indicate evaluation context window sizes.

Model	# Param.	256		480		Inference Time(ms/t)
		Val.	Test	Val.	Test	
Adaptive Inputs w/ LN	247M	19.5	20.2	19.3	20.0	13.9
Adaptive Inputs w/ PRepBN (Ours)	247M	**19.2**	**20.0**	**19.1**	**19.8**	**12.9**

Table 5. Experimental results of the proposed method for LLaMA-350M on various benchmarks.

Model	Throughput	ARC-C	ARC-E	BoolQ	COPA	HellaSwag	PIQA	WinoGrande	OpenBookQA	SciQ	Avg.
LLaMA-350M	44.0	22.95	46.13	59.27	64	33.19	64.36	49.09	29.6	75.3	49.32
w/ PRepBN (Ours)	**50.4**	23.55	44.44	60.67	66	32.76	64.04	49.72	30.0	76.8	**49.78**

Table 6. Ablation studies for the impact of simplified linear attention and progressive re-parameterized BatchNorm.

Method	FLOPs	Lat. (ms)	Acc. (%)
Flatten-DeiT-T	1.1 G	15.2	74.1
+ SLA	1.1 G	10.2	73.0
+ SLA + PRepBN	1.1 G	**9.6**	**74.3**
Flatten-PVT-T	2.0 G	10.8	77.8
+ SLA	2.0 G	8.5	75.2
+ SLA + PRepBN	2.0 G	**8.0**	**76.5**
Flatten-Swin-T	4.5 G	10.9	82.1
+ SLA	4.5 G	9.5	81.9
+ SLA + PRepBN	4.5 G	**8.7**	**81.8**
Flatten-Swin-S	8.8 G	18.6	83.5
+ SLA	8.8 G	18.0	83.4
+ SLA + PRepBN	8.7 G	**16.2**	**83.6**

5.4. Ablation Studies

In this section, we conduct extensive ablation studies to demonstrate the impact of our key designs.

The impact of SLA and PRepBN. We first explore the impact of the simplified linear attention (SLA) module and progressive re-parameterized BatchNorm (PRepBN) on different backbones. As shown in Table 6, utilizing our simplified linear attention (SLA) brings consistent improvement for efficiency. For DeiT and PVT, our SLA obtains significant latency reduction and a few accuracy drop. Moreover, Swin transformers equipped with our SLA achieve quite comparable accuracy with that of original ones but with lower latency. In addition, the latency could be further reduced by replacing LayerNorm by our proposed progressive re-parameterized BatchNorm (PRepBN). This strategy hardly affects the accuracy and even recover the accuracy of model like DeiT and PVT. Combining these two strategies, the latency is reduced by 5.6 ms when the accuracy is improved by 0.2% for DeiT-T. Moreover, our method obtains similar accuracy and harvests 2.2 ms and 2.4 ms latency reduction for Swin-T and Swin-S models.

Ablation study for PRepBN. We investigate key components of our proposed PRepBN, *i.e.*, the progressive strategy and re-parameterized BatchNorm (RepBN). Directly training a BatchNorm-based transformer leads to quite unstable training, either obtaining inferior performance or collapse in training (*e.g.*, DeiT-S and Flatten-Swin-T). To avoid the variance shift (Li et al., 2019) caused by droppath, which will influence the performance of BatchNorm, we simply set the droppath rate to 0 on DeiT-T model. As shown in Table 7, applying progressive strategy on a BatchNorm-based DeiT-T model brings 1.2% accuracy gain. We further utilize our RepBN in the model and the accuracy increases to 73.6%. These results demonstrate that both our proposed progressive strategy and re-parameterized BatchNorm (RepBN) are beneficial for training a pure BatchNorm-based transformer.

Table 7. Ablation studies for the impact of progressive strategy and re-parameterized BatchNorm.

Method	Acc. (%)
DeiT-T-BN	71.9
+ Progressive Strategy	73.1
+ Progressive Strategy + RepBN	**73.6**

6. Conclusion

In this paper, we investigates the computational bottleneck modules of transformer and propose novel strategies including progressive Re-parameterized BatchNorm and simplified linear attention to obtain efficient transformer architectures. Our method progressively replace LayerNorm with re-parameterized BatchNorm during training to obtain lossless accuracy, while leveraging the efficiency advantages of BatchNorm during inference. Additionally, we devise a simplified linear attention mechanism that attains comparable performance with other linear attention methods but with less computational cost. Through extensive experiments for both computer vision and language modeling tasks, we showcase that our method achieves stronger performance with respect to accuracy and efficiency than prior methods and sheds light into the design of efficient transformer.

Acknowledgments. We gratefully acknowledge the support of MindSpore (Huawei, 2020), CANN (Compute Architecture for Neural Networks) and Ascend AI Processor used for this research.

Impact Statement

This paper presents work whose goal is to advance the field of Deep Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Baevski, A. and Auli, M. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.

Bolya, D., Fu, C.-Y., Dai, X., Zhang, P., and Hoffman, J. Hydra attention: Efficient attention with many heads. In *European Conference on Computer Vision*, pp. 35–49. Springer, 2022.

Cai, H., Gan, C., and Han, S. Efficientvit: Enhanced linear attention for high-resolution low-computation visual recognition. *arXiv preprint arXiv:2205.14756*, 2022.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *European conference on computer vision*, pp. 213–229. Springer, 2020.

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., and Guo, B. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12124–12134, 2022.

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.

Han, D., Pan, X., Han, Y., Song, S., and Huang, G. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5961–5971, 2023.

Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):87–110, 2022.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

He, W., Han, K., Tang, Y., Wang, C., Yang, Y., Guo, T., and Wang, Y. Densemamba: State space models with dense hidden connection for efficient large language models. *arXiv preprint arXiv:2403.00818*, 2024.

Huawei. Mindspore. https://www.mindspore.cn/, 2020.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Larsson, G., Maire, M., and Shakhnarovich, G. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

Li, X., Chen, S., Hu, X., and Yang, J. Understanding the disharmony between dropout and batch normalization by variance shift. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2682–2690, 2019.

Li, Z., Xiao, J., Yang, L., and Gu, Q. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17227–17236, 2023.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Tang, Q., Liu, C., Liu, F., Liu, Y., Jiang, J., Zhang, B., Han, K., and Wang, Y. Category feature transformer for semantic segmentation. *arXiv preprint arXiv:2308.05581*, 2023a.

Tang, Q., Zhang, B., Liu, J., Liu, F., and Liu, Y. Dynamic token pruning in plain vision transformers for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 777–786, 2023b.

Tang, Y., Wang, Y., Guo, J., Tu, Z., Han, K., Hu, H., and Tao, D. A survey on transformer compression. *arXiv preprint arXiv:2402.05964*, 2024.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Tu, Z., Talebi, H., Zhang, H., Yang, F., Milanfar, P., Bovik, A., and Li, Y. Maxvit: Multi-axis vision transformer. In *European conference on computer vision*, pp. 459–479. Springer, 2022.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., and Shao, L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 568–578, 2021.

Wang, Y., Chen, X., Cao, L., Huang, W., Sun, F., and Wang, Y. Multimodal token fusion for vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12186–12195, 2022.

Wu, X., Zeng, F., Wang, X., and Chen, X. PPT: Token pruning and pooling for efficient vision transformers. *arXiv preprint arXiv:2310.01812*, 2023.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Xia, Z., Pan, X., Song, S., Li, L. E., and Huang, G. Dat++: Spatially dynamic vision transformer with deformable attention. *arXiv preprint arXiv:2309.01430*, 2023.

Xu, Y., Li, C., Li, D., Sheng, X., Jiang, F., Tian, L., and Sirasao, A. Fdvit: Improve the hierarchical architecture of vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5950–5960, 2023.

Yan, J., Wan, R., Zhang, X., Zhang, W., Wei, Y., and Sun, J. Towards stabilizing batch statistics in backward propagation of batch normalization. *arXiv preprint arXiv:2001.06838*, 2020.

Yang, Q., Zhang, K., Lan, C., Yang, Z., Li, Z., Tan, W., Xiao, J., and Pu, S. Unified normalization for accelerating and stabilizing transformers. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 4445–4455, 2022.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Yao, Z., Cao, Y., Lin, Y., Liu, Z., Zhang, Z., and Hu, H. Leveraging batch normalization for vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 413–422. IEEE, 2021.

Zheng, D., Dong, W., Hu, H., Chen, X., and Wang, Y. Less is more: Focus attention for efficient detr. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6674–6683, 2023.

Zhu, L., Wang, X., Ke, Z., Zhang, W., and Lau, R. W. Biformer: Vision transformer with bi-level routing attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10323–10333, 2023.

A. Detailed hyper-parameter settings.

The detailed hyper-parameter settings are provided in Table A1. For image classification, we follow the setting of DeiT (Touvron et al., 2021). We use AdamW as our default optimizer and train all of model for 300 epochs with a cosine decay learning rate scheduler and 20 epochs of linear warm-up. The batch size is set to 1024, an initial learning rate is 0.001, and the value of weight decay is 0.05. For model used proposed progressive re-parameterized BatchNorm, we employ an smaller rate of droppath (Larsson et al., 2016). Owing to the variance shift caused by droppath, We freeze the model parameters and only update the statistics of re-parameterized BatchNorm for 10 epochs at the end of training.

Table A1. The settings of hyperparameters on different models for image classification task.

Name	**DeiT-T**	**DeiT-S**	**PVT-T**	**PVT-S**	**PVT-M**	**Swin-T**	**Swin-S**	**Swin-B**
Epoch	300	300	300	300	300	300	300	300
Batch Size	1024	1024	1024	1024	1024	1024	1024	1024
Learning rate	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
Warmup Steps	20	20	20	20	20	20	20	20
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Droppath Rate	0.0	0.0	0.0	0.1	0.3	0.1	0.3	0.3
Linear Decay Steps	0	3e5	0	3e5	3e5	0	3e5	3e5

B. Combination with post-quantization method.

Our method focuses on replacing LayerNorm with BatchNorm to obtain inference speed-up without performance degradation. It is a complementary strategy with common model compression methods like weight quantization, pruning or distillation and these methods can be combined to achieve a better performance. As a proof of concept, we conduct post-quantization using RepQ-ViT (Li et al., 2023) on DeiT-Tiny model that is trained with our PRepBN strategy. As shown in the below table, applying W8A8 quantization on top of our method still achieves the accuracy of 73.6%, while further reducing the computational cost to only 0.33 GFLOPs. This demonstrates that our method can be effectively combined with other compression methods such as quantification.

Table A2. The settings of hyperparameters on different models for image classification task.

Method	**FLOPs (G)**	**Throughput (images/s)**	**Top-1 Acc (%)**
DeiT-T	1.3	3432	72.2
w/ PRepBN (Ours)	1.3	4194	**73.6**
w/ PRepBN (Ours) + RepQ-ViT (W8A8)	0.33	-	**73.6**