NETBOOSTER: EMPOWERING TINY DEEP LEARNING BY STANDING ON THE SHOULDERS OF DEEP GIANTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tiny deep learning has attracted increasingly growing interest driven by the substantial demand for deep learning solutions in numerous Internet-of-Things (IoT) applications. Nevertheless, due to the under-fitting issue, it is still a challenge to unleash tiny deep learning's full potential on large-scale datasets. Consequently, tiny neural networks' (TNNs') downstream task performance is limited due to the inferior learned representations during pretraining. To this end, we propose a framework dubbed NetBooster which empowers tiny deep learning from a novel perspective by augmenting the architecture of TNNs via an expansion-then-contraction strategy. Specifically, during training, our proposed NetBooster first expands each/some layer(s) of a given TNN into multi-layer blocks, favoring the learning of more complex features to generate an expanded counterpart model (i.e., deep giant), and then contracts the expanded layers by gradually removing the non-linear layers from the expanded ones to recover efficiency. NetBooster's expansion-then-contraction training empowers its trained TNNs to benefit from the superior performance of their expanded counterparts while preserving the TNNs' original complexity and thus inference efficiency. Extensive experiments and ablation studies on two tasks, seven datasets, and six networks validate that NetBooster consistently leads to a nontrivial accuracy boost (e.g., 1.3% $\sim$ 2.5%) on top of state-of-the-art TNNs on ImageNet and as much as 4.7% higher accuracy on various downstream datasets, while maintaining their inference complexity/efficiency.

1 INTRODUCTION

Powerful deep learning networks are often accompanied by prohibitively huge computational and memory costs (Liu et al., 2021c; Brown et al., 2020; He et al., 2016; Liu et al., 2021b), hindering their wider applications in resource-constrained Internet-of-Things (IoT) devices (Lin et al., 2020). To this end, tiny deep learning, which aims to develop tiny neural networks (TNNs) featuring a much-reduced network size along with memory and computational costs to enable deep learning-powered solutions in tiny IoT devices (e.g., MCU (Lin et al., 2020) and Raspi (Raspberry Pi Limited.)), has emerged as a promising direction and attracted an increasingly growing interest from both industry and academia (Lin et al., 2020; Cai et al., 2021). In particular, existing tiny deep learning works strive to improve the performance of TNNs by either designing a novel efficient network architecture manually or automatically (Lin et al., 2020; 2021; Sandler et al., 2018; Howard et al., 2019), or compressing a large deep neural network (DNN) to reduce its network redundancy via DNN compression techniques, such as pruning, quantization, etc (Liu et al., 2017; Fu et al., 2021a; Wang et al., 2020; Teerapittayanon et al., 2016; He et al., 2018; Liu et al., 2020).

Despite both the extensive efforts and the promising progress in developing novel TNN architectures and compressing existing DNN networks, the achieved accuracy-efficiency trade-off of existing TNN works is still far from satisfactory for many IoT and emerging applications. Specifically, TNNs' accuracy-efficiency trade-off bottleneck comes from two folds: ***Iss. 1***: It is challenging for TNNs to learn complex but representative features and achieve satisfactory task performance on commonly used large-scale datasets (e.g., ImageNet (Deng et al., 2009)), and ***Iss. 2***: TNNs' limited task performance on large-scale datasets further hinders TNN-based solutions from taking advantage of the widely used pretrain-then-finetune paradigm in real-world deployment.

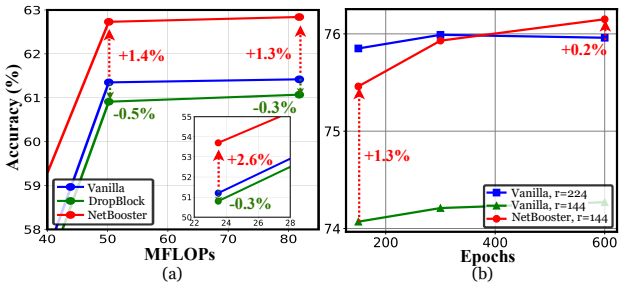

Figure 1: (a) ***Iss. 1***: TNN training suffers from under-fitting issues. When training MobileNetV2 (Sandler et al., 2018) on ImageNet (Deng et al., 2009), regularization techniques (e.g., DropBlock (Ghiasi et al., 2018)) even lead to inferior accuracy compared with vanilla training. Our proposed NetBooster can boost TNNs' accuracy by increasing its capacity during training. (b) ***Iss. 2***: Inadequately trained TNNs cannot learn complex features and thus suffer from limited downstream task accuracy. Finetuning ImageNet pretrained vanilla MobileNetV2-35 with a resolution of 224×224 and 144×144, respectively, on the CIFAR-100 dataset for even four times more epochs (i.e., 600 epochs) still cannot improve the achievable accuracy. Our proposed NetBooster can boost TNNs' accuracy by inheriting pretrained deep giants' learned complex features.

In parallel, it has recently been recognized that a dedicated training recipe can boost the accuracy of TNNs (Cai et al., 2021), which yet is still under-explored. Unlike DNN training, which requires techniques like data augmentation (Hendrycks et al., 2019; Zhang et al., 2017; Cubuk et al., 2020; 2018) and/or regularization (Ghiasi et al., 2018; Srivastava et al., 2014) to alleviate the *over-fitting* issue, recent studies (Tan & Le, 2019; Cai et al., 2021) have shown that TNNs tend to suffer from *under-fitting* issues due to their small capacity, and extensively augmented training data or regularized training can even hurt the achieved task performance of TNNs. This is because TNNs have a limited ability to learn complex features from the given large-scale dataset during training (i.e., ***Iss. 1***). For example, applying the state-of-the-art (SOTA) regularization technique introduced in DropBlock(Ghiasi et al., 2018) to ResNet-50 (He et al., 2016) can lead to a 1.6% accuracy improvement on the challenging ImageNet dataset (Deng et al., 2009); however, as shown in Fig. 1 (a), training MobileNetV2 (Sandler et al., 2018) of various sizes with DropBlock (green line) even results in inferior task performance (an accuracy drop of $0.5\% \sim 0.7\%$) compared with the corresponding vanilla trained networks (blue line). Moreover, when transferring pretrained TNNs to downstream tasks, the lack of learned complex and representative features in the pretrained TNN models further limits the achievable accuracy of downstream tasks, which cannot be recovered even with more finetuning epochs (i.e., ***Iss. 2***). For instance, as shown in Fig. 1 (b), transferring a vanilla ImageNet (Deng et al., 2009) pretrained MobileNetV2-35 (Cai et al., 2018) to CIFAR-100 (Krizhevsky et al., 2009) only leads to an accuracy of 75.8% and 74.07% with an input resolution of 224×224 (blue line) and 144×144 (green line), respectively, and even adopting four times more training epochs cannot improve the accuracy due to the inferior learned features during pretraining.

To close the gap between the growing demand for more powerful TNNs in real-world applications and the lack of effective TNN training schemes, we aim to develop techniques that can boost the achievable task performance of TNNs, while maintaining their attractive efficiency, by empowering TNNs' learnable features. In particular, this work makes the following contributions:

- To the best of our knowledge, we are the first to discover and promote a new paradigm of training TNNs to empower their achievable accuracy via constructing a competent deep giant with compound network augmentation (i.e., augmenting both width and depth dimensions of the given TNNs), which is simple, effective, and generally applicable.
- Leveraging the above discovery, we propose a TNN training framework dubbed NetBooster that can alleviate TNNs' under-fitting issue during training and thus boost their achievable accuracy, while maintaining their original network complexity and thus inference efficiency. Specifically, our NetBooster integrates a two-step expansion-then-contraction training strategy: **Step-1: Network Expansion** to expand each/some layer(s) of a given TNN into multi-layer blocks, favoring the learning of more complex features by leveraging the corresponding deep giant counterpart to equip the TNN with an initial state that have learned sufficient knowledge, and **Step-2: Progressive Linearization Tuning (PLT)** to convert the deep giant back to the given TNN's original architecture by removing the non-linear layers from the expanded layers and then contracting them.
- We make heuristic efforts to empirically explore (1) what kinds of multi-layer blocks should be used for expansion, (2) when to expand during training, and (3) where to expand within

a TNN to more effectively boost the achieved accuracy of TNNs when implementing compound network augmentation in our proposed NetBooster framework.

- Extensive experiments and ablation studies on two tasks, six networks, and seven datasets show that NetBooster consistently leads to a nontrivial accuracy boost (e.g., 1.3% $\sim$ 2.6%) on top of SOTA TNNs on the ImageNet dataset and as much as 4.7% higher accuracy on various downstream tasks, while maintaining inference complexity/efficiency.

2 Related Works

2.1 Tiny Neural Network

Tiny deep learning aims to develop TNNs featuring a much reduced network size and lower memory and computational costs, together with acceptable accuracy, enabling deep learning-powered solutions in resource-constrained IoT devices. Existing techniques towards fulfilling the goal of tiny deep learning can mostly be categorized into two trends. One trend is to design novel TNN architectures by resorting to either human expertise (Sandler et al., 2018; Ma et al., 2018; Zhang et al., 2018) or automated tools, e.g., neural architecture search (Cai et al., 2018; Wu et al., 2019; Wan et al., 2020; Fu et al., 2021b); The other trend is to make use of compression techniques, including pruning (Liu et al., 2018; 2017; Li et al., 2021), quantization (Fu et al., 2021a; Zhou et al., 2017), dynamic inference (Wang et al., 2020; Wu et al., 2018; Teerapittayanon et al., 2016; Yu et al., 2021), etc., to further trim down network complexity on top of previously designed TNN architectures.

In this work, we propose to pursue tiny deep learning solutions with boosted performance-efficiency trade-offs from a less explored and orthogonal direction, i.e., how to train TNNs to unleash their achievable accuracy more effectively. To the best of our knowledge, the only pioneering work focusing on a similar direction is NetAug (Cai et al., 2021), which proposes to augment TNNs from the width dimension by introducing a wider supernet to assist training and then directly removing the supernet. In contrast, our NetBooster proposes to first expands a TNN network *from both the depth and width dimensions* to provide a competent deep giant during TNN training, and then gradually contract it back to the original structure, *instead of directly removing augmented parts*, to avoid unrecoverable information losses which might incur nontrivial accuracy drops.

2.2 Data augmentation and regularization

Data augmentation and regularization techniques have been proposed to alleviate the over-fitting issues associated with large-scale DNNs in order to boost their network generalization performance and thus achievable accuracy. Data augmentation techniques focus on manipulating the input data samples (Cubuk et al., 2018; Liu et al., 2018; Zhong et al., 2020; Gong et al., 2021) while regularization techniques focus more on the network aspect and randomly drop different components from the network (Ghiasi et al., 2018; Srivastava et al., 2014; Huang et al., 2016) during training.

Nevertheless, as shown in recent studies (Tan & Le, 2019; Cai et al., 2021) and Fig. 1 (a), TNN training suffers from under-fitting instead of over-fitting. Existing data augmentation and regularization techniques fail to fully unleash the potential of TNNs.

2.3 Knowledge Distillation

Knowledge distillation (KD) aims to transfer the already learned knowledge to new training processes by leveraging an often larger teacher network to guide a smaller student network (Hinton et al., 2015; Zagoruyko & Komodakis, 2016; Tung & Mori, 2019; Huang & Wang, 2017; Park et al., 2019; Heo et al., 2019).

Our proposed NetBooster is orthogonal to KD, and it does not require a teacher network to provide guidance during training. Instead, NetBooster aims to inherit the learned features from the expanded TNNs and thus can achieve higher accuracy than the original network. As such, it is expected that combining our proposed NetBooster with existing KD techniques can further boost TNNs' performance.

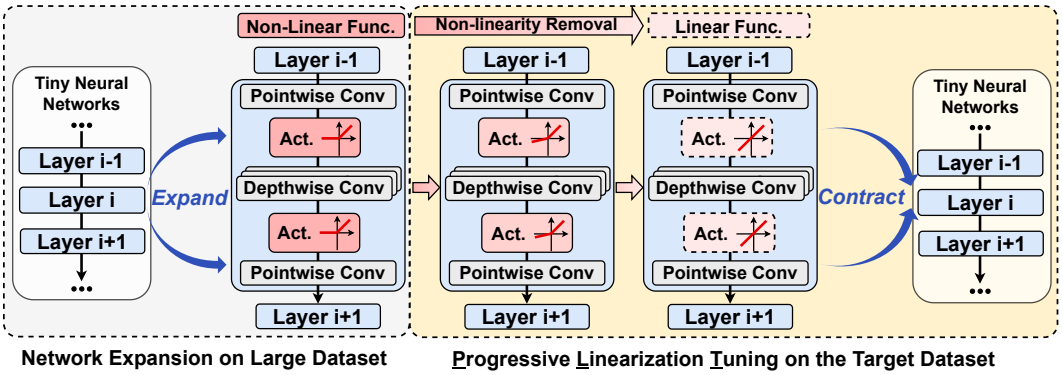

Figure 2: An overview of the proposed NetBooster framework. In **NetBooster**, we augment the TNN from both depth and width dimensions. Specifically, we uniformly select TNN layers and expand them into inverted residual blocks Sandler et al. (2018) to formulate the deep giant, helping to learn complex features. Then in PLT, we progressively decay the non-linear activation functions within the expanded inverted residual blocks to an identity mapping function and contract the expanded blocks back to the corresponding original layers to maintain the given TNN's original structure and inference efficiency.

2.4 Transfer Learning

Motivated by DNNs' strong capability in extracting features, using transfer learning (Donahue et al., 2014; Chatfield et al., 2014; Sharif Razavian et al., 2014; Azizpour et al., 2015) under the pretrain-then-finetune paradigm to inherit pretrained DNNs' representative and generalizable features for downstream tasks is a widely used paradigm across different domains (Mormont et al., 2018; Lim, 2012; Houlsby et al., 2019). Despite the extensive efforts to boost the transfer performance, it is still a common understanding that larger pretrained networks can provide more representative and generalizable features and thus lead to better transfer task performance than training TNNs from scratch. Our proposed NetBooster aims to narrow the aforementioned gap by empowering TNNs with high-quality features and achieves this by leveraging TNNs' corresponding deep giants through compound network augmentation.

3 The Proposed NetBooster Framework

3.1 Motivations and Inspirations

The key challenge for TNN training. Due to the lack of sufficient network capacity, TNNs tend to suffer from severe under-fitting issues when being trained on large-scale datasets (e.g., ImageNet (Deng et al., 2009)), limiting their ability to learn complex but representative features and further hindering their achievable performance on downstream tasks.

Inspirations for our works. Recent works show that overparameterization during training can benefit the final achievable accuracy while the network complexity during inference can be trimmed down without hurting the accuracy (Liu et al., 2021a). Different DNN compression techniques have also echoed this insight, e.g., pruning methods (Han et al., 2015; Liu et al., 2017) keep the original dense network during training and then remove the redundant neurons for inference. Nevertheless, the aforementioned methods only focus on overparameterization from the width dimension. At the same time, existing work (Nguyen et al., 2020) shows that DNNs with different depth and width tend to learn different features, urging for introducing overparameterization into both depth and width dimensions. Thus, if we can equip TNNs with comprehensive overparameterization during training and then restore the original network structure during inference, the under-fitting issue can be effectively mitigated to achieve more accurate yet efficient TNN inference. This has inspired us to design a principled expansion-then-contraction methodology by first expanding the target TNN to a more overparameterized network for better feature learning and then contract it back to the original structure to enhance its efficiency.

3.2 OVERVIEW

Implementation of expansion-then-contraction. Given the expansion-then-contraction principle, there are different potential implementations. Inspired by the success of RepVGG (Ding et al., 2021), which shows that parallel branches can be merged thanks to their linearity, we hypothesize that if we can properly remove the non-linear activation functions, the consecutive layers can also be merged via a linear combination. Fortunately, recent works show that some of the activation functions can be safely removed from the network without hurting the task performance (Jha et al., 2021; Cho et al., 2021; Ghodsi et al., 2021). This motivates us to propose our expansion-then-contraction-based NetBooster training framework, which adopts two steps: **Step-1: Network Expansion**, where we augment both TNNs' depth and width during training by replacing some layers in the original TNNs with multi-layer blocks, aiming to increase the TNNs' capacity and alleviate their under-fitting issues during training and thus enabling a better feature learning on the source dataset, and **Step-2: Progressive Linearization Tuning (PLT)**, where we progressively remove the non-linearity *inside* the expanded blocks on the target dataset. After the non-linear layers inside the blocks are removed, we contract the augmented/expanded TNNs back to the corresponding original TNNs at the end of training to inherit the learned features while ensuring the boosted performance does not come with additional inference cost.

Technical challenges to achieve NetBooster. While the aforementioned principle sounds straightforward, implementing such a training pipeline is non-trivial. In **Step-1**, naively expanding all layers with a high expansion ratio can lead to an excessively large network which is difficult to train. To enable a practical and effective expansion strategy, at least the following three questions need to be addressed: $\mathcal{Q}1$: what blocks to insert? What kind of blocks we should use to expand the target layer, $\mathcal{Q}2$: where to expand? How to find which layers we should expand, and $\mathcal{Q}3$: how to determine the expansion ratio? To what extent we should expand the target layers. In **Step-2**, how to contract the network expanded from both width and depth dimensions while preserving the learned knowledge of the augmented network is still an open question. Despite the method proposed in (Ding et al., 2021) can merge the parallel convolution layers into one single layer via linear combination, the non-linear activation layers between sequentially connected convolution layers make it impossible to merge convolution layers along the depth dimension directly. We next elaborate on our proposed solutions to tackle the above challenges and the design of each step as follows.

3.3 STEP 1: NETWORK EXPANSION

The network expansion step aims to transform the target TNNs into more powerful deep giants, boosting the network capacity and learning complex but representative features from the large-scale dataset to improve the accuracy and transferability. To answer the questions raised in Sec. 3.2, we propose the following criteria when expanding the network:

- **a. Structural consistency:** To guarantee that applying NetBooster does not change the network structure for inference, each expansion block will be required to contract back to the original single layer via linear combination in the PLT step. Thus, for the network expansion step, the receptive field of each expansion block should be equal to that of the original layer.
- **b. Sufficient capacity:** Motivated by the findings in (Cai et al., 2021), we aim to alleviate the under-fitting issue and ease the learning process by increasing the capacity of the expanded network (i.e., the corresponding deep giant). Thus, we should sufficiently expand the given TNN from multiple positions and dimensions (i.e., expand width with increased expansion ratios and depth by inserting multiple layers).
- **c. Effective feature inheritance:** In addition to the sufficient capacity of the deep giant, effectively inheriting the deep giant's learned features is equally important. As suggested in (Mirzadeh et al., 2020), excessively large networks tend to learn significantly different feature distribution from that of small networks, which can not only forbid small networks from inheriting but even hurt the small networks' task performance. Thus, (1) the complexity gap between the original network and its expanded deep giant should not be too large and (2) the selected layers to be expanded should contain sufficient parameters to ensure an effective knowledge inheritance from the deep giant.

Based on the above criteria, we answer the questions raised in Sec. 3.2 below:

$\mathcal{Q}1$ What blocks to insert? We select the types of inserted blocks from a pool of well-established DNN building blocks (e.g., the basic and bottleneck blocks in ResNet (He et al., 2016) and the inverted residual block in MobileNetV2 (Sandler et al., 2018)). The basic block is first eliminated to satisfy structure consistency (criteria **a.**). As the basic block stacks two layers with large convolution kernels, leading to a receptive field larger than that of the original layer, we then select the inverted residual block over the bottleneck block to narrow down the complexity gap for effective feature inheritance (criteria **c.**).

$\mathcal{Q}2$ Where to expand? There is a trade-off between increasing the network capacity by constructing a larger deep giant (criteria **b.**) and improving the feature inheritance effectiveness by narrowing down the size gap between the deep giant and original TNN (criteria **c.**), limiting the achievable performance of NetBooster. A simple but effective way to push forward the trade-off further is to consider the knowledge inheritance effectiveness from a more fine-grained granularity (i.e., layer-wise) instead of model-wise. Specifically, multiple layers can have a better representation ability than a single layer. Thus, the expanded layer's learned complex features can be more effectively inherited by distributing them to multiple adjacent layers in the contracted network. To this end, we propose uniformly selecting layers to be expanded on top of the original network, which can guarantee that there are sufficient layers to inherit learned features from the expanded block.

$\mathcal{Q}3$ How to determine expansion ratio? Similar to $\mathcal{Q}2$, the selection of the adopted expansion ratio has to trade-off between the network capacity (criteria **b.**) and the effectiveness of knowledge inheritance (criteria **c.**). However, thanks to the proposed uniform expansion strategy in $\mathcal{Q}2$, we empirically find that the commonly used expansion ratio, 6 (Sandler et al., 2018) in the inserted inverted residual blocks works well on balancing the aforementioned two criteria

3.4 Step 2: Progressive Linearization Tuning (PLT)

The next step is to recover the original network structure on the target dataset while inheriting the knowledge learned by the deep giant. Inspired by (Ding et al., 2021), which proposes to merge parallel convolution layers into one single layer, we find that sequentially connected layers can also be merged via linear combinations by properly removing the non-linear operations between them. To achieve this, we propose **PLT** to progressively remove the non-linearity from the expanded network and then contract the expanded network back to its original architecture during finetuning on the target dataset.

Motivating observation. Non-linearity has been considered a key enabler for the promising performance of DNNs and most existing works use the combination of convolution and non-linear activation layers as a basic design unit. In parallel, recent works (Jha et al., 2021; Cho et al., 2021) have shown that **non-linearity within DNNs can be highly redundant for inference**, a large portion of element-wise non-linear activation functions can be removed from a DNN, and the complex features learned from the original network during training can be largely preserved. Inspired by the revolution from element-wise pruning to structure pruning, **we aim to step further and remove the non-linearity in a structured manner (i.e., layerwisely)**.

Non-linearity removal. We propose to transform the expanded deep giant back to the original network meanwhile preserve the learned features learned by slowly decaying the non-linear activation functions.

Without loss of generality, we take the ReLU activation function as an example as it is the most commonly adopted activation function in TNNs, and our following discussions can also be extended to other activation functions like ReLU6. Here the ReLU function is defined as:

$$Y_l = \max(0, X_l), \tag{1}$$

where X_l and Y_l are the input and output of layer l, respectively. We change the formulation of ReLU to the following format,

$$Y_l = \max(\alpha_l X_l, X_l), \tag{2}$$

where $0 < \alpha_l < 1$ is the slope parameter to manipulate the non-linearity of the corresponding activation layer. When $\alpha_l = 0$, it is exactly the ReLU function. When $\alpha_l = 1$, the activation function is decayed to an identity mapping.

Given a list L of non-linear activation layers to be removed, we increase $\alpha_{l'}$ for $l' \in L$ from 0 to 1 in E_d epochs, the value of $\alpha_{l'}$ is uniformly increased in each iteration. When $\alpha_{l'} = 1$, Eq. 2 is an identity mapping, and thus the non-linearity is removed.

Expanded block contraction. With the non-linear activation functions removed, the remaining layers can be contracted into one layer via simple linear combinations.

Formulation: Without lose of generality, we take two convolution layers as an example. Given the input to the first layer $X \in \mathcal{R}^{h_1 \times w_1 \times c_1}$, the output $Y \in \mathcal{R}^{h_3 \times w_3 \times c_3}$, as well as the kernels of two layers $K^1 \in \mathcal{R}^{k_1 \times k_1 \times c_1 \times c_2}$ and $K^2 \in \mathcal{R}^{k_2 \times k_2 \times c_2 \times c_3}$, the overall functionality of the two convolution layers can be formulated as

$$Y_{p,q,o} = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \sum_{m=0}^{c_1-1} X_{p-i,q-j,m} K_{i,j,m,o}, \tag{3}$$

$$where\ K_{i,j,m,o} = \sum_{s=s_l}^{s_h} \sum_{t=t_l}^{t_h} \sum_{n=0}^{c_2-1} K^1_{i-s,j-t,m,n} K^2_{s,t,n,o}, \tag{4}$$

where $k = k_1 + k_2 - 1$, $s_l = \max(0, i - k_1 + 1)$, $s_h = \min(k_2 - 1, i)$, $t_l = \max(0, j - k_2 + 1)$ and $t_h = \min(k_2 - 1, j)$.

Remark: It is worth noting that different expansion ratios of the inserted inverted residual block will result in the same computational cost after contraction since the input and output channels after contraction are always equal to the input channel of the first layer and the output channel of the last layer, respectively, regardless of intermediate layers.

4 Experiments

4.1 Experiments Setup

Tasks, datasets, and networks We consider two tasks, including image classification and object detection, with seven datasets to provide a thorough evaluation of NetBooster. Specifically, to evaluate NetBooster's performance in alleviating the under-fitting performance to achieve higher accuracy on the large-scale dataset, we consider the ImageNet dataset (Deng et al., 2009). To evaluate how deep giant's learned better representation helps with downstream tasks, we consider image classification task on five datasets, including CIFAR-100 (Krizhevsky et al., 2009), Cars (Krause et al., 2013), Flowers102 (Nilsback & Zisserman, 2008), Food101 (Bossard et al., 2014), and Pets (Parkhi et al., 2012). We also evaluate the downstream object detection task performance on the Pascal VOC dataset (Everingham et al., 2010). We consider six networks, including ResNet-50 (He et al., 2016), different variants of MobileNet (Sandler et al., 2018), including MobileNetV2-100/50/Tiny, neural architecture searched hardware friendly network MCUNet (Lin et al., 2020), and recently trending vision transformer DeiT-Tiny (Touvron et al., 2021).

Baselines. We benchmark the proposed NetBooster over vanilla networks under standard training, a series of SOTA KD algorithms (i.e., tf-KD (Yuan et al., 2020), RCO-KD (Jin et al., 2019), and RocketLaunch (Zhou et al., 2018)), and NetAug (Cai et al., 2021), which is the pioneering work aiming at boosting TNN training performance.

Expansion strategy. We uniformly expand 50% of blocks in the target TNN. To expand each block, we replace the first pointwise convolution layer with an inverted residual block with an expansion ratio of 6. The kernel size of the depthwise convolution layer in the inserted inverted residual block is set to 1 to make the receptive field the same as the original pointwise convolution.

Training settings. We develop our training settings based on the commonly adopted settings. Specifically, when evaluating NetBooster's performance in improving TNNs' performance on large-scale ImageNet dataset, we follow (Cai et al., 2021) to train the deep giant (expanded network) on ImageNet for 160 epochs using SGD optimizer with a batch size of 1024, an initial learning rate of 0.2 and cosine anneal learning rate schedule. In PLT, we set $E_d = 40$, and further finetune for 110 epochs also on ImageNet. For downstream tasks transfer, we use the ImageNet pretrained deep giant (expanded network) as the starting point and develop our training recipe on CIFAR-100 based

on (Tian et al., 2019), on Cars, Flowers102, Food101, and Pets based on (Salman et al., 2020), and on Pascal VOC based on (Cai et al., 2021). In all experiments on downstream tasks, we assign E_d to 20% of the total tuning epochs to PLT and finetune the network in the remaining epochs.

4.2 MITIGATING ***Iss. 1***: BENCHMARKING ON LARGE-SCALE DATASET

To evaluate whether the proposed NetBooster can help TNNs to learn the complex features on the large-scale dataset and thus improve performance on the challenging dataset, we benchmark NetBooster on ImageNet dataset with NetAug and various of KD algorithms. As shown in Table 1, NetBooster achieves 1.3% $\sim$ 2.5% accuracy improvements over the vanilla networks, showing the strong ability to boost the TNN accuracy by standing on the shoulder of deep giants generated by NetBooster.

Table 1: Benchmarking on **ImageNet**. 'r' is the input resolution.

Network	FLOPs	Params	Training Method	Accuracy
MobileNetV2-Tiny (r=144)	23.5M	0.75M	Vanilla	51.2
			RocketLaunch (Zhou et al., 2018)	51.8
			tf-KD (Yuan et al., 2020)	51.9
			RCO-KD (Jin et al., 2019)	52.6
			NetAug (Cai et al., 2021)	53.0
			NetBooster	**53.7**
MCUNet (r=176)	81.8M	0.74M	Vanilla	61.4
			NetAug (Cai et al., 2021)	62.5
			NetBooster	**62.8**
MobileNetV2-50 (r=160)	50.2M	1.95M	Vanilla	61.4
			NetAug (Cai et al., 2021)	62.5
			NetBooster	**62.7**
MobileNetV2-100 (r=160)	154.1M	3.47M	Vanilla	69.6
			NetAug (Cai et al., 2021)	70.5
			NetBooster	**70.9**

Compared with the KD baselines, our proposed NetBooster achieves 0.9% $\sim$ 1.1% accuracy improvement without guidance from the teacher network (Assemble-ResNet50 (Lee et al., 2020)), suggesting that the network expansion in NetBooster can equip the expanded network with sufficient capacity to effectively learn the complex features at least comparable with the large-scale teacher DNN used in the KD baselines, proving the learned features can be effectively inherited by the target network from the PLT step. Moreover, enabling training without a teacher network in NetBooster further leads to only 32.5% additional iteration-wise training latency over vanilla training, which is 19.4% less than vanilla KD.

Compared with NetAug, which is a pioneering work focusing on a similar scenario as NetBooster, we also achieve consistently superior performance over their method, suggesting the multi-dimensional network expansion and the PTL for features inheritance is more effective than the network width expansion and directly drop augmented neuron after training proposed in NetAug. It is worth noticing that, even training NetAug with two times of training epochs, NetBooster still achieves 0.3% higher accuracy than NetAug (53.7% v.s. 53.4%). Please refer to Appendix A for NetBooster's performance on more models.

Table 2: Benchmarking on **downstream image classification datasets**. 'r' is the input resolution.

Network	Training Method	CIFAR-100	Cars	Flowers102	Food101	Pets
MobileNetV2-Tiny (r=144)	Vanilla	74.07	76.18	90.01	75.43	78.30
	NetBooster	75.46	80.93	90.53	75.96	78.90
MobileNetV2-35 (r=160)	Vanilla	76.08	78.36	90.63	76.80	80.64
	Vanilla + KD	76.38	77.47	91.41	77.02	82.44
	NetBooster	76.66	80.91	91.16	77.26	80.92
	NetBooster + KD	77.15	83.36	92.68	77.81	83.37

4.3 MITIGATING ***Iss. 2***: BENCHMARKING ON DOWNSTREAM TASKS

To evaluate whether the learned complex and representative features in deep giant from the large-scale dataset can further help TNNs to achieve better downstream tasks performance, we first evaluate NetBooster's performance when transferring the ImageNet pretrained deep giant to five representative smaller-scale image classification datasets with PLT. As shown in Table 2, compared with vanilla training, NetBooster's framework achieves 0.46% $\sim$ 4.75% accuracy improvement, showing the complex features learned by a deep giant are effectively inherited after PLT, leading to higher downstream task performance. Our method is also orthogonal to KD, where applying KD together with our proposed NetBooster can lead to another 0.49% $\sim$ 2.45% accuracy boost on top of NetBooster alone.

Table 3: Benchmarking on **object detection** tasks with Pascal VOC dataset with MobileNetV2-35 at 416 resolution.

Method	Vanilla	NetAug	NetBooster
AP50	60.8	62.4	**62.6**

We further evaluate NetBooster's performance when transferring to the Pascal VOC object detection task. As shown in Table 3, NetBooster achieves 1.8 and 0.2 higher AP50 compared with vanilla

training and NetAug, respectively. This proves that NetBooster can be considered as a general method to boost TNNs' performance across various tasks.

4.4 VALIDATING EXPANSION STRATEGY

We validate our answer to each question in Sec. 3.3 by validating the impact of replacing our proposed strategy with alternatives when training MobileNet-Tiny on the ImageNet dataset with an input resolution of 144.

Table 4: Ablation study on **what kind of block** to insert.

Inserted Block Type	Expanded Acc.	Final Acc.
Vanilla	-	51.20
Inverted Residual	54.90	53.70
Basic Block	54.52	53.41
Bottleneck Block	55.23	53.62

Q1. What blocks to insert: We ablate the impact of expanding with different kinds of blocks and report the results in Table 4. Expanding with inverted residual blocks leads to slightly better results ($0.29\% \sim 0.08\%$). Showing (1) the NetBooster framework can robustly boost TNNs' performance, and (2) inserting with the inverted residual block is an effective choice.

Table 5: Ablation study on **which block** to expand.

Expansion	Expanded FLOPs	Expanded Params	Expanded Acc.	Final Acc.
Vanilla	29.4M	0.75M	-	51.20
Expand First 8	65.0M	0.83M	51.46	51.50
Expand Middle 8	49.6M	0.93M	52.98	52.62
Expand Last 8	51.2M	1.25M	53.90	52.47
Uniform Expand 8	63.9M	0.99M	54.90	53.70

Q2. Where to expand: We ablate different expansion locations' impacts and report our findings in Table 5. We observe that uniformly expanding the model achieves $2.20\% \sim 1.08\%$ higher accuracy compared with excessively expanding the first/middle/last part of the network, proving the necessity to expand the model uniformly.

Q3. How to determine expansion ratio: We ablate different selection of expansion ratios in the inserted inverted residual blocks and report the results in Table. 6. We observe that NetBooster with a wide range of commonly used expansion ratios (i.e., $4 \sim 6$) consistently improves the TNNs' performance, further proving NetBooster's robustness to hyperparameter selection.

Please refer to Appendix B for more ablation study on our proposed expansion strategy.

4.5 SCHEDULE FOR NON-LINEARITY REMOVAL

Table 6: Ablation study on **expansion ratio** of inserted block.

Expansion ratio	2	4	6	8
Final Acc.	52.94	53.52	53.70	52.56

One of the key factors to NetBooster's success is to minimize the knowledge loss after progressively removing the non-linearity in inserted blocks during PLT. Thus, we further ablate the appropriate way to remove the non-linearity from the network by training the MCUNet on the ImageNet dataset with different E_d. We summarize the results in Table 7. Surprisingly, as long as we use the proposed PLT to decay the non-linearity gradually, the variance in decay epochs only leads to a marginal change in the achieved accuracy (i.e., less than 0.3% accuracy change), proving the necessity and the stability of PLT.

5 CONCLUSION

Table 7: Ablation study on **non-linearity decay epochs**.

Network	E_d	Accuracy
MCUNet	0	61.8
	20	62.6
	40	62.8
	60	62.5

In this paper, we discover and promote a new paradigm for training TNNs to empower their achievable accuracy via augmenting both dimensions of the network (i.e., depth and width) during training. Furthermore, we propose a framework dubbed NetBooster, which is dedicated to boosting the performance of SOTA TNNs by using an expand-then-contract training strategy to alleviate TNNs' underfitting issue. Finally, we make heuristic efforts to empirically explore what/when/where to augment when training TNNs using our proposed NetBooster. Extensive experiments and ablation studies on seven networks and two datasets show that NetBooster consistently leads to a nontrivial accuracy boost (e.g., $1.3\% \sim 2.5\%$) on top of SOTA TNNs on the ImageNet and as much as 4.7% higher accuracy on various downstream tasks, while maintaining their inference complexity/efficiency.

REFERENCES

Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. IEEE transactions on pattern analysis and machine intelligence, 38(9):1790–1802, 2015.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In European conference on computer vision, pp. 446–461. Springer, 2014.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. arXiv preprint arXiv:2005.14165, 2020.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332, 2018.

Han Cai, Chuang Gan, Ji Lin, and Song Han. Network augmentation for tiny deep learning. arXiv preprint arXiv:2110.08890, 2021.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531, 2014.

Minsu Cho, Zahra Ghodsi, Brandon Reagen, Siddharth Garg, and Chinmay Hegde. Sphynx: Relu-efficient network design for private inference. arXiv preprint arXiv:2106.11755, 2021.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501, 2018.

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255. Ieee, 2009.

Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13733–13742, 2021.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In International conference on machine learning, pp. 647–655. PMLR, 2014.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2): 303–338, 2010.

Yonggan Fu, Qixuan Yu, Meng Li, Vikas Chandra, and Yingyan Lin. Double-win quant: Aggressively winning robustness of quantized deep neural networks via random precision training and inference. In International Conference on Machine Learning, pp. 3492–3504. PMLR, 2021a.

Yonggan Fu, Zhongzhi Yu, Yongan Zhang, Yifan Jiang, Chaojian Li, Yongyuan Liang, Mingchao Jiang, Zhangyang Wang, and Yingyan Lin. Instantnet: Automated generation and deployment of instantaneously switchable-precision networks. arXiv preprint arXiv:2104.10853, 2021b.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. arXiv preprint arXiv:1810.12890, 2018.

Zahra Ghodsi, Nandan Kumar Jha, Brandon Reagen, and Siddharth Garg. Circa: Stochastic relus for private deep learning. Advances in Neural Information Processing Systems, 34, 2021.

Chengyue Gong, Dilin Wang, Meng Li, Vikas Chandra, and Qiang Liu. Keepaugment: A simple information-preserving data augmentation approach. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1055–1064, 2021.

Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. arXiv preprint arXiv:1506.02626, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European conference on computer vision (ECCV), pp. 784–800, 2018.

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781, 2019.

Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pp. 3779–3787, 2019.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In International Conference on Machine Learning, pp. 2790–2799. PMLR, 2019.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1314–1324, 2019.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In European conference on computer vision, pp. 646–661. Springer, 2016.

Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. arXiv preprint arXiv:1707.01219, 2017.

Nandan Kumar Jha, Zahra Ghodsi, Siddharth Garg, and Brandon Reagen. Deepreduce: Relu reduction for fast private inference. In International Conference on Machine Learning, pp. 4839–4849. PMLR, 2021.

Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1345–1354, 2019.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In Proceedings of the IEEE international conference on computer vision workshops, pp. 554–561, 2013.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Jungkyu Lee, Taeryun Won, Tae Kwan Lee, Hyemin Lee, Geonmo Gu, and Kiho Hong. Compounding the performance improvements of assembled techniques in a convolutional neural network. arXiv preprint arXiv:2001.06268, 2020.

Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8607–8617, 2021.

Joseph Jaewhan Lim. Transfer learning by borrowing examples for multiclass object detection. PhD thesis, Massachusetts Institute of Technology, 2012.

Ji Lin, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, and Song Han. Mcunet: Tiny deep learning on iot devices. arXiv preprint arXiv:2007.10319, 2020.

Ji Lin, Wei-Ming Chen, Han Cai, Chuang Gan, and Song Han. Mcunetv2: Memory-efficient patch-based inference for tiny deep learning. arXiv preprint arXiv:2110.15352, 2021.

Ning Liu, Xiaolong Ma, Zhiyuan Xu, Yanzhi Wang, Jian Tang, and Jieping Ye. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 4876–4883, 2020.

Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? in-time over-parameterization in sparse training. In International Conference on Machine Learning, pp. 6989–7000. PMLR, 2021a.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. arXiv preprint arXiv:2111.09883, 2021b.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030, 2021c.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE international conference on computer vision, pp. 2736–2744, 2017.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. arXiv preprint arXiv:1810.05270, 2018.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European conference on computer vision (ECCV), pp. 116–131, 2018.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pp. 5191–5198, 2020.

Romain Mormont, Pierre Geurts, and Raphaël Marée. Comparison of deep transfer learning strategies for digital pathology. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 2262–2271, 2018.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. arXiv preprint arXiv:2010.15327, 2020.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing, pp. 722–729. IEEE, 2008.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3967–3976, 2019.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In 2012 IEEE conference on computer vision and pattern recognition, pp. 3498–3505. IEEE, 2012.

Raspberry Pi Limited. Raspberry Pi 4. `https://www.raspberrypi.org/products/raspberry-pi-4-model-b/`, accessed 2020-09-01.

Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? Advances in Neural Information Processing Systems, 33:3533–3545, 2020.

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

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 806–813, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.

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

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 2464–2469. IEEE, 2016.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. arXiv preprint arXiv:1910.10699, 2019.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In International Conference on Machine Learning, pp. 10347–10357. PMLR, 2021.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1365–1374, 2019.

Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12965–12974, 2020.

Yue Wang, Jianghao Shen, Ting-Kuei Hu, Pengfei Xu, Tan Nguyen, Richard Baraniuk, Zhangyang Wang, and Yingyan Lin. Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference. IEEE Journal of Selected Topics in Signal Processing, 14(4):623–633, 2020.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10734–10742, 2019.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8817–8826, 2018.

Zhongzhi Yu, Yonggan Fu, Sicheng Li, Chaojian Li, and Yingyan Lin. Mia-former: Efficient and robust vision transformers via multi-grained input-adaptation. arXiv preprint arXiv:2112.11542, 2021.

Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3903–3911, 2020.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928, 2016.

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

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 6848–6856, 2018.

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

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. arXiv preprint arXiv:1702.03044, 2017.

Guorui Zhou, Ying Fan, Runpeng Cui, Weijie Bian, Xiaoqiang Zhu, and Kun Gai. Rocket launching: A universal and efficient framework for training well-performing light net. In Thirty-second AAAI conference on artificial intelligence, 2018.

A NetBooster as a General Technique to Improve Network Performance

Existing works have shown that although larger networks normally suffer from over-fitting instead of under-fitting, increasing the network capacity can still lead to non-trivially performance improvement (Liu et al., 2021c; He et al., 2016; Liu et al., 2021b). To validate whether NetBooster has the potential as a general technique to boost neural networks' performance, we further validate the NetBooster on ResNet-50 (He et al., 2016) and DeiT-Tiny (Touvron et al., 2021) on the ImageNet dataset with input resolution 224.

Table 8: Evaluating NetBooster's performance on ResNet-50 and DeiT-Tiny.

Network	Training method	Accuracy
ResNet-50	Vanilla	77.6
	NetBooster	**78.0**
DeiT-Tiny	Vanilla	72.2
	NetBooster	**73.7**

Experiments setting: We follow the training setting in (He et al., 2016) and (Touvron et al., 2021) to train ResNet-50 and DeiT-Tiny, respectively. We follow the proposed setting to expand 50% of blocks in ResNet-50 and DeiT-Tiny uniformly. Specifically, for ResNet-50, we expand the first layer into the inverted residual block with a depthwise convolution kernel size of 3, expansion ratio 6 in the first layer in the selected blocks. For DeiT-Tiny, we expand the first fully connected layer in the feedforward network of the selected blocks into two fully connected layers with an expansion ratio of 6. We set E_d to 20% of the total training epochs.

Results: As shown in Table 8, NetBooster leads to 0.4% higher accuracy over vanilla training on the large-scale DNN (i.e., ResNet-50) and 1.5% higher accuracy over the attention-based vision transformer (i.e., DeiT-Tiny), showing its promising potential in boosting all DNNs' performance.

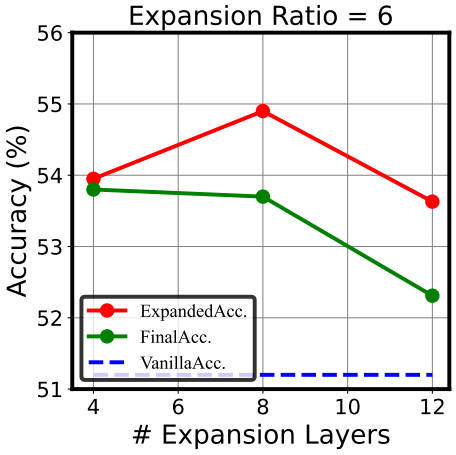

Figure 3: Visualization of the expanded accuracy (i.e., the accuracy of the expanded network) and final accuracy when using different numbers of expand blocks with an expansion ratio 6.

B Number of Expansion Layers

The other factor that controls the deep giant's capacity is the number of expansion layers. We evaluate the relationship between the expansion ratio and the final performance on MobileNetV2-Tiny and the ImageNet dataset with an input resolution of 144. As shown in Fig. 3, we observe that with various numbers of expansion layers, (1) networks boosted by the expansion-then-contraction strategy in NetBooster can constantly surpass the performance of vanilla networks, proving the NetBooster's effectiveness, (2) the expanded network performance (ExpandAcc.) increase first then decrease. We suspect the decrease in the expanded network accuracy is due to the overly deep network makes it challenging to train the network effectively, and (3) the boosted

network performance (FinalAcc.) also has the increase-then-decrease pattern, validating our selected number of expansion layers.