GhostNetV3: Exploring the Training Strategies for Compact Models

Zhenhua Liu[1⋆], Zhiwei Hao[2⋆], Kai Han[1⋆⋆], Yehui Tang[1], and Yunhe Wang[1⋆⋆]

[1] Huawei Noah's Ark Lab, Beijing, China
{liu.zhenhua,kai.han,yehui.tang,yunhe.wang}@huawei.com
[2] Beijing Institute of Technology, Beijing, China
haozhw@bit.edu.cn

Abstract. Compact neural networks are specially designed for applications on edge devices with faster inference speed yet modest performance. However, training strategies of compact models are borrowed from that of conventional models at present, which ignores their difference in model capacity and thus may impede the performance of compact models. In this paper, by systematically investigating the impact of different training ingredients, we introduce a strong training strategy for compact models. We find that the appropriate designs of re-parameterization and knowledge distillation are crucial for training high-performance compact models, while some commonly used data augmentations for training conventional models, such as Mixup and CutMix, lead to worse performance. Our experiments on ImageNet-1K dataset demonstrate that our specialized training strategy for compact models is applicable to various architectures, including GhostNetV3, MobileNetV2 and ShuffleNetV2. Specifically, equipped with our strategy, GhostNetV3 $1.3\times$ achieves a top-1 accuracy of 79.1% with only 269M FLOPs and a latency of 14.46ms on mobile devices, surpassing its ordinarily trained counterpart by a large margin. Moreover, our observation can also be extended to object detection scenarios. PyTorch code and checkpoints can be found at https://github.com/huawei-noah/Efficient-AI-Backbones.

1 Introduction

To meet the limited memory and computational resource of edge devices (*e.g.*, mobile phone), various efficient architectures [10, 15, 16, 19, 21, 26, 28] have been developed. For example, MobileNetV1 [16] uses depth-wise separable convolutions to reduce computational cost. MobileNetV2 [21] introduces the residual connection, and MobileNetV3 [15] further optimizes the architecture configurations via neural architecture search (NAS), which significantly improves performance of the model. Another typical architecture is GhostNet [10], which utilizes redundancy in feature and duplicates channels of the feature by using

---

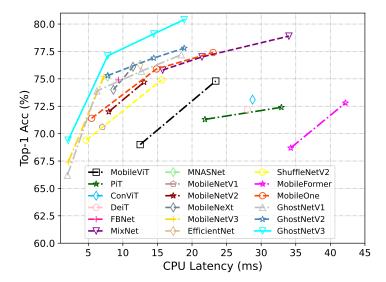⋆ Equal contribution
⋆⋆ Corresponding author

Fig. 1: The top-1 validation accuracy and the latency on CPU of various compact models on ImageNet dataset.

cheap operations. Recently, GhostNetV2 [26] further incorporates a hardware-friendly attention module to capture the dependence between long-range pixels and outperforms GhostNet by a significant margin.

Besides carefully designed model architectures, appropriate training strategies are also critical for remarkable performance. For example, Wightman *et al.* [29] improved the top-1 accuracy of ResNet-50 [11] on ImageNet-1K [5] from 76.1% to 80.4% by integrating advanced optimization and data augmentation approaches. However, although considerable efforts have been made to explore more advance training strategies for conventional models (*e.g.*, ResNet and Vision Transformer), little attention has been paid to compact models. Since models with different capacities may have different learning preferences [24], directly applying strategies designed for conventional models to train compact models is not appropriate.

To bridge this gap, we systematically investigate several training strategies for compact models. Specifically, our main attentions are paid on the key training settings as discussed in previous works [12, 27] including re-parameterization, knowledge distillation (KD), learning schedule and data augmentation.

Re-parameterization. Depth-wise convolution and 1×1 convolution are common components in compact model architectures due to their negligible memory and computational consumption. Inspired by successful experiences in training conventional models [6, 7], we employ the re-parameterization approach for these two compact modules to achieve better performance. When training compact models, we introduce linear parallel branches into depth-wise

convolution and 1×1 convolution. These additional parallel branches can be re-parameterized after training, bringing no extra cost at inference time. To trade off the overall training cost against performance improvement, we compare the impact of different numbers of added branches. Moreover, we find that a 1×1 depth-wise convolution branch has a significant positive impact on the re-parameterization of 3×3 depth-wise convolution.

Knowledge distillation. It is challenging for compact models to achieve performance comparable to conventional models due to their limited model capacity. Hence, KD [14], which employs a larger model as the teacher to guide the learning of compact models, is a proper approach to improve performance. We empirically investigate the impact of several typical factors when training compact models using KD, such as the choice of teacher model and the setting of hyperparameters. The results imply that an appropriate teacher model can significantly boost the performance of compact models.

Learning schedule and data augmentation. We compare several training settings for compact models, including learning rate, weight decay, exponential moving average (EMA), and data augmentation. Interestingly, not all the tricks designed for conventional models work well for compact models. For instance, some widely-used data augmentation methods, such as *Mixup* and *CutMix*, actually detract from the performance of compact models. We discuss their effect in Section 5 in detail.

Based on our investigation, we develop a specialized training recipe for compact models. Experiments on ImageNet-1K dataset verify the superiority of our proposed recipe. Specifically, the GhostNetV3 model trained with our recipe significant outperforms that trained with the previous strategy in terms of both top-1 accuracy and latency (Figure 1). Experiments on other efficient architectures such as MobileNetV2 and ShuffleNetV2 further confirm the generalizability of the proposed recipe.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 introduces the architecture of GhostNetV2. The training strategies are discussed in detail in Section 4. Then the plentiful experimental results are presented in Section 5. Section 6 concludes the paper.

2   Related works

2.1   Compact models

It is challenging to design a compact model architecture with low inference latency and high performance simultaneously. SqueezeNet [17] proposes three strategies to design a compact model, i.e., replacing 3×3 filters with 1×1 filers, decreasing the number of input channels to 3×3 filters, and down-sampling late in the network to keep large feature maps. These principles are constructive, especially the usage of 1×1 convolution. MobileNetV1 [16] replaces almost all the filers with 1×1 kernel and depth-wise separable convolutions, which dramatically reduces the computational cost. MobileNetV2 [21] further introduces the

residual connection to the compact model, and constructs an inverted residual structure, where the intermediate layer of a block has more channels than its input and output. To keep representation ability, a part of non-linear functions are removed. MobileNeXt [36] rethinks the necessary of inverted bottleneck, and claims that the classic bottleneck structure can also achieve high performance. Considering the 1×1 convolution account for a substantial part of computational cost, ShuffleNet [34] replace it with group convolution. The channel shuffle operation to help the information flowing across different groups. By investigating the factors that affect the practical running speed, ShuffleNetV2 [18] proposes a new hardware-friendly block.

MnasNet [22] and MobileNetV3 [15] search the architecture parameters, such as model width, model depth, convolutional filter's size, *etc*. By leveraging the feature's redundancy, GhostNet [10] replaces half channels in 1×1 convolution with cheap operations. GhostNetV2 [26] proposes a dubbed DFC attention based on fully-connected layers, which can not only execute fast on common hardware but also capture the dependence between long-range pixels. Until now, the series of GhostNets are still the SOTA compact models with a good trade-off between accuracy and speed.

Since ViT [8] (DeiT) has made a great success on computer vision tasks, researchers have made efforts to design compact transformer architecture for mobile devices. MobileFormer [2] proposes a compact cross attention to model the two-way bridge between MobileNet and transformer. MobileViT [19] takes the successful experiences in compact CNNs and replaces local processing in convolutions with global processing using transformers. However, the transformer-based compact models suffer from high inference latency on mobile devices due to the complex attention operation.
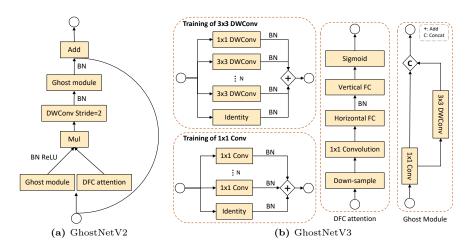

Fig. 2: The architectures of GhostNetV2 and GhostNetV3.

2.2   Bag of tricks for training CNNs

There are some works that focusing on improving training strategies to improve the performance of various models. He *et al.* [12] discuss several tricks that are useful for efficient training on hardware and propose a new model architecture tweaks for ResNet. Wrightman *et al.* [29] re-evaluate the performance of the vanilla ResNet-50 when trained with novel optimization and data-augmentation methods. They share the competitive training settings and pre-trained model in the timm open-source library. With their training recipe, a vanilla ResNet-50 model achieves 80.4% top-1 accuracy. Chen *et al.* [1] investigate the effects of several fundamental components for training self-supervised ViT. However, all these attempts are designed for large models or self-supervised models. Directly transferring them to compact models is inappropriate because of their different model capacity [24].

3   Preliminary

GhostNets (GhostNetV1 and GhostNetV2) are the state-of-the-art compact models designed for efficient inference on mobile devices. Their key architecture is the Ghost module, which can replace the original convolution by generating more feature maps from cheap operations.

In the ordinary convolution, the output feature Y is obtained by $Y = X * W$, where $W \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k \times k}$ is the convolution kernel and X is the input feature. c_{in} and c_{out} denote the input and output channel dimension, respectively. k is the kernel size and $*$ denotes convolution operation. A Ghost module reduces the number of parameters and computational cost of ordinary convolution in two steps. It first produces *intrinsic* features Y', whose channel dimension is smaller than the original feature Y. Then the cheap operation (*e.g.*, depth-wise convolution) is applied on the intrinsic features Y' to generate *ghost* features Y''. The final output is obtained by concatenating the *intrinsic* and *ghost* features along the channel dimension, which can be formulated as:

$$Y' = X * W_p, \tag{1}$$
$$Y = \text{Cat}(Y', X * W_c), \tag{2}$$

where W_p and W_c denote the parameters in primary convolution and cheap operations, respectively. "Cat" denotes the concatenating operation. A whole GhostNet model is constructed by stacking multiple Ghost modules.

GhostNetV2 enhances compact models by designing an efficient attention module, *i.e.*, DFC attention. Considering that compact models such as Ghost-Net usually use small-kernel convolutions, *e.g.*, 1×1 and 3×3, they have weak ability to extract global information from input features. GhostNetV2 employs a simple fully-connected layer to capture the long-range spatial information and generate an attention map. For computational efficiency, it decouples the global information into horizontal and vertical directions and aggregates pixels along

the two directions, respectively. As shown in Figure 2a, by equipping Ghost module with DFC attention, GhostNetV2 can extract global and local information effectively while achieves a better trade-off between accuracy and computational complexity.

4   Training strategies

Our goal is to exploring the training strategies without changing the inference network architectures to keep the small model size and fast speed of compact models. We empirically investigate the key factors for training neural networks including learning schedule, data augmentation, re-parameterization and knowledge distillation.

4.1   Re-parameterization

Re-parameterization has proved its effectiveness in conventional convolutional models [6, 7]. Inspired by their success, we introduce re-parameterization into compact models by adding repetitive branches equipped with BatchNorm layers. Our design of re-parameterization GhostNetV3 is presented in Figure 2b. It is worth noting that we introduce a 1×1 depth-wise convolution branch into re-parameterized 3×3 depth-wise convolution. Experimental results confirm its positive effect on the performance of compact models. Furthermore, the experiments thoroughly explore the optimal number of repetitive branches.

At inference time, the repetitive branches can be removed via an inverse re-parameterization process. Since convolution and BatchNorm operations are both linear during inference, they can be folded into a single convolution layer, whose weight matrix is denoted as $\widehat{\mathbf{W}} \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times k \times k}$ and bias is denoted as $\widehat{\mathbf{b}} \in \mathbb{R}^{c_{\text{out}}}$. After that, folded weights and biases in all branches can be re-parameterized into $\mathbf{W}_{\text{rep}} = \sum_i \widehat{\mathbf{W}}_i$ and bias $\mathbf{b}_{\text{rep}} = \sum_i \widehat{\mathbf{b}}_i$, respectively, where i is the index of repetitive branches.

4.2   Knowledge distillation

KD is a widely used model compression method, where the predictions of a large pre-trained teacher model is regarded as the learning target of a tiny student model. Given a sample x with label y, representing the corresponding logits predicted by the student and the teacher model using $\Gamma_{\text{s}}(x)$ and $\Gamma_{\text{t}}(x)$, respectively, the total loss function of KD can be formulated as:

$$\mathcal{L}_{\text{total}} = (1 - \alpha)\mathcal{L}_{\text{ce}}(\Gamma_{\text{s}}(x), y) + \alpha\mathcal{L}_{\text{kd}}(\Gamma_{\text{s}}(x), \Gamma_{\text{t}}(x)), \qquad (3)$$

where $\mathcal{L}_{\text{ce}}$ and $\mathcal{L}_{\text{kd}}$ denote the cross-entropy loss and KD loss respectively. α is a balancing hyperparameter.

Usually, the Kullback-Leibler divergence function is adopted as the KD loss, which can be represented as:

$$\mathcal{L}_{\text{kd}} = \tau^2 \cdot \text{KL}(\text{softmax}(\Gamma_{\text{s}}(x))/\tau, \text{softmax}(\Gamma_{\text{t}}(x))/\tau), \qquad (4)$$

where τ is a label smoothing hyperparameter termed temperature. In our experiments, we study the impact of different settings of hyperparameters α and τ on the performance of compact models.

4.3 Learning schedule

Learning rate is a critical parameter in the optimization of neural networks. There are two commonly used learning rate schedules: *step* and *cosine*. The *step* schedule decreases the learning rate linearly, whereas *cosine* schedule reduces the learning rate slowly at the beginning, becomes almost linear in the middle, and slows down again at the end. This work extensively investigates the impact of both learning rate and learning rate schedule on compact models.

Exponential moving average (EMA) has recently emerged as an effective approach to improve the validation accuracy and increase the robustness of models. Specifically, it gradually averages parameters of a model during the training time. Suppose parameters of the model at step t is $\mathbf{W}_t$, the EMA of the model is computed as:

$$\overline{\mathbf{W}}_t = \beta \cdot \overline{\mathbf{W}}_{t-1} + (1 - \beta) \cdot \mathbf{W}_t, \tag{5}$$

where $\overline{\mathbf{W}}_t$ represents the parameters of the EMA model at step t and β is a hyperparameter. We study the effect of EMA in Section 5.3.

4.4 Data augmentation

Various data augmentation approaches have been proposed to promote performance of conventional models. Among them, *AutoAug* scheme [3] adopts 25 combinations of sub-strategy, each of which contains two transformations. For each input image, a sub-strategy combination is randomly selected, and the decision of whether to apply each transformation in the sub-strategy is determined by a certain probability. *RandomAug* method [4] proposes a random augmentation approach where all sub-strategies are selected with the same probability. Image aliasing methods like *Mixup* [33] and *CutMix* [32] fuse two images to generate a new image. Specifically, *Mixup* trains a neural network on convex combinations of pairs of examples and their labels, whereas *CutMix* randomly removes a region from one image and replace the corresponding area with a patch from another image. *RandomErasing* [35] randomly selects a rectangle region in an image and replaces its pixels with random values.

In this paper, we evaluate various combinations of the above data augmentation approaches and find that some commonly used data augmentation methods for training conventional models, such as *Mixup* and *CutMix*, are not appropriate for training compact models.

5 Experimental results

In our basic training strategy, we use a mini-batch size of 2048 and employ LAMB [31] for model optimization over 600 epochs. The initial learning rate

Table 1: Top-1 accuracy of different versions of GhostNetV3 models trained with or without re-parameterization.

Table 2: Comparison of top-1 accuracy for different values of re-parameterization factor N. 'DW' denotes depth-wise convolution.

Model size	1.0×	1.3×	1.6×
w/o Rep	75.3	76.9	77.8
w/ ReP	76.1	77.6	78.7

Number of branches	w/o 1×1DW	w/ 1×1DW
N=1	77.3	77.0
N=2	77.2	77.3
N=3	77.1	77.6
N=4	77.0	77.5
N=5	77.0	77.6

is 0.005, and the *cosine* learning schedule is adopted. Weight decay and momentum are set to 0.05 and 0.9, respectively. We use a decay factor of 0.9999 for the exponential moving average (EMA), where random augmentation and random erasing are applied for data augmentation. In this section, we explore these training strategies and unveil insights for training compact models. All experiments are conducted on the ImageNet dataset [5] using 8 NVIDIA Tesla V100 GPUs.

5.1   Re-parameterization

To better comprehend the advantages of integrating re-parameterization into the training of compact models, we perform an ablation study to assess the impact of re-parameterization on different sizes of GhostNetV3. The results are presented in Table 1. The adoption of re-parameterization, while keeping other training settings unchanged, results in a significant improvement in performance compared to directly training the original GhostNetV3 models.

Additionally, we compare different configurations of the re-parameterization factor N and the results are presented in Table 2. As indicated in the results, the 1×1 depth-wise convolution plays a crucial role in re-parameterization. If the 1×1 depth-wise convolution is not used in the re-parameterized model, its performance even decreases as the number of branches increases. In contrast, when equipped with a 1×1 depth-wise convolution, the GhostNetV3 model achieves a peak top-1 accuracy of 77.6% when N is 3, and further increasing the value of N brings no additional improvement in performance. Therefore, the re-parameterization factor N is set to 3 in subsequent experiments to achieve better performance.

5.2   Knowledge distillation

In this section, we assess the impact of knowledge distillation on the performance of GhostNetV3. Specifically, ResNet-101 [11], DeiT-B [27], and BeiTV2-B [20] are adopted as the teacher, achieving top-1 accuracies of 77.4%, 81.8%, and 86.5%, respectively. The results in Table 3 highlight performance variations with different teacher models. Notably, superior teacher performance correlates

Table 3: Comparison of Top-1 accuracy for various teachers and α in knowledge distillation.

Teacher \ α	0.5	0.9	1.0
ResNet-101 [11]	78.46	78.53	78.20
DeiT-B [27]	78.61	78.55	78.32
BEiTV2-B [20]	79.13	79.02	78.86

Table 4: Comparison of Top-1 accuracy for different α and temperature τ in knowledge distillation.

α \ τ	1	2	4
0.5	79.13	77.98	77.91
1.0	78.94	77.92	77.70

Table 5: Comparison of Top-1 accuracy for combining re-parameterization and knowledge distillation.

Number of branches	1	2	3
w/o 1×1DW	78.59	78.73	78.85
w/ 1×1DW	78.80	79.04	79.13

Table 6: Top-1 accuracy of different learning rate schedule for GhostNetV2. The number after 'cosine' denotes the minimum of the learning rate.

lr schedule	step	cosine_0	cosine_1e-5
w ReP&KD	78.40	79.13	78.97

with improved GhostNetV3 performance, underscoring the importance of a well-performing teacher model in knowledge distillation with compact models.

We additionally compare different settings of hyperparameters in the KD loss with BEiTV2-B as the teacher. The results in Table 4 suggest that a low temperature value is preferable for compact models. Furthermore, it is worth noting that when the KD loss is used alone (*i.e.* $\alpha=1.0$), there is a noticeable decline in the top-1 accuracy.

We also explore the impact of combining re-parameterization and knowledge distillation on the performance of GhostNetV3. As shown in Table 5, the results indicate a significant improvement in performance (up to 79.13%) due to the utilization of knowledge distillation. Moreover, it emphasizes the importance of the 1×1 depth-wise convolution in re-parameterization. These findings underscore the importance of investigating various techniques and their potential combinations to enhance the performance of compact models.

5.3   Learning schedule

Learning rate schedule. Figure 3 illustrates the experimental results adopting different scheduling schemes of the learning rate, both with and without re-parameterization and knowledge distillation. It is observed that both small and large learning rates have a detrimental effect on performance. Therefore, a learning rate of 5e-3 is chosen for the final experiments.

The *step* and *cosine* learning rate schedules are compared in Table 6. It is observed that the *cosine* learning rate schedule achieves the highest top-1 accuracy. This underlines the effectiveness of a well-designed *cosine* learning rate schedule in promoting the performance of compact models.
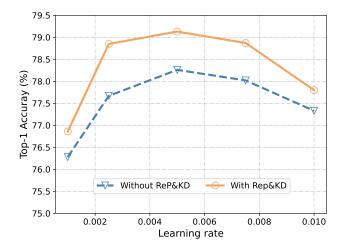
Fig. 3: The top-1 validation accuracy for various learning rates of GhostNetV3.

Weight decay. The effect of weight decay on the top-1 accuracy of GhostNetV3 is shown in Table 7. The results indicate that a large weight decay significantly diminishes the model's performance. Therefore, we have retained a weight decay value of 0.05 for GhostNetV3, given its effectiveness for compact models.

Table 7: Top-1 accuracy achieved with different weight decay settings on the ImageNet dataset.

Weight decay	0.01	0.03	0.05	0.07	0.1
w ReP&KD	78.63	78.85	79.13	79.01	78.83

EMA. In Figure 4, it can be observed that when the decay of EMA is 0.99999, there is a decline in performance regardless of whether re-parameterization and knowledge distillation techniques are used or not. We speculate that this is due to the weakening effect of the current iteration when the decay value is too large. For compact models, a decay value of 0.9999 or 0.99995 is deemed appropriate, which is similar to that of a conventional model.

5.4   Data Augmentation

To compare the impact of different data augmentation schemes on the performance of lightweight models, we train cnn-based GhostNetV3 and ViT-based
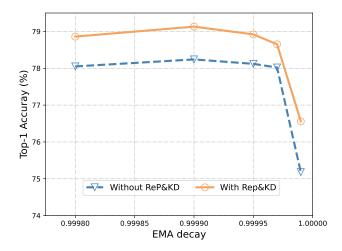
Fig. 4: The top-1 accuracy achieved with various decay values of EMA.

DeiT-tiny models under distinct augmentation strategies. The results are presented in Table 8. It is observed that *random augmentation* and *random erasing* are advantageous for both GhostNetV3 and DeiT-tiny. Conversely, *Mixup* and *CutMix* have a detrimental effect, which are then considered unsuitable for compact models.

Table 8: comparison results of different combinations of data augmentation schemes on GhostNetV3 and DeiT-tiny.

AutoAug	RandAug	Mixup	CutMix	Erasing	GhostNetV3	DeiT-tiny
✓	✗	✗	✗	✗	78.78	76.33
✗	✓	✗	✗	✗	79.03	76.30
✗	✓	✓	✗	✗	78.95	76.23
✗	✓	✓	✓	✗	78.48	76.11
✗	✓	✓	✓	✓	78.41	76.02
✗	✓	✗	✗	✓	**79.13**	**76.38**

5.5   Comparison with other compact models

In this section, we compare GhostNetV3 with other compact models in terms of parameters, FLOPs, latency on CPU and mobile phone. Specifically, we run the models on a Windows desktop equipped with a 3.2GHz Intel i7-8700 processor to measure the CPU latency and use a Huawei Mate40Pro equipped with a Kirin
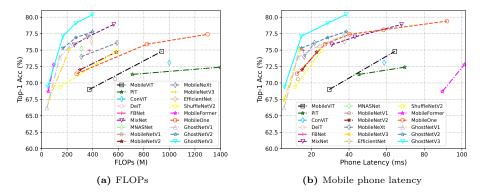
(a) FLOPs   **(b)** Mobile phone latency

Fig. 5: The FLOPs and the latency of the compact models on mobile phone.

9000 CPU to evaluate the mobile phone latency under the configuration of input resolution of 224×224. To ensure the lowest latency and the highest consistency, all other applications on both the CPU and mobile phone are closed. Each model is executed for 100 times to obtain reliable results.

Table 9 provides a detailed comparison of GhostNetV3 with other compact models, whose parameter counts are under 20M. From the results, the smallest transformer-based architectures require a latency of 12.5ms for inference on mobile devices, while their top-1 accuracy is only 69.0%. In contrast, GhostNetV3 achieves a top-1 accuracy of 77.1% with a significantly lower latency of 7.81ms. The current state-of-the-art model, MobileFormer [2], achieves a top-1 accuracy of 79.3% with a latency of 129.58ms, which is unaffordable in real-world applications. In comparison, GhostNetV3 1.6× achieves better accuracy of 80.4% with a significantly lower latency of 18.87ms, which is 6.8× faster than MobileFormer.

Next, we compare GhostNetV3 with other CNN-based compact models, including MobileNets [15, 16, 21, 36], ShuffleNets [18, 34], MixNet [25], MNAS-Net [22], FBNet [30], EfficientNet [23], and MobileOne [28], where FBNet, MNAS-Net, and MobileNetV3 are search-based models and the others are manually designed models. Specifically, FBNet employs a hardware searching strategy while MNASNet and MobileNetV3 search the architecture parameters, such as model width, model depth, the size of the convolutional filter, *etc.*

Compared with MobileNetV2 [21], GhostNetV3 1.0× achieves a 5.1% improvement while maintaining almost the same latency (7.81ms vs. 7.96ms). GhostNetV3 1.3× also exhibits improved top-1 accuracy compared to MobileNeXt and EfficientNet-B0, which are 3.0% and 2.8%, respectively. In particular, when compared with the powerful manually designed MobileOne models, GhostNetV3 1.0× outperforms MobileOne-S1 in terms of top-1 accuracy by 1.2% with only half the latency required. GhostNetV3 1.3× also outperforms MobileOne-S2 by 1.7% while costing only 60% of the latency. Moreover, when GhostNet 1.6× achieves a higher top-1 accuracy than MobileOne-S4 (80.4% vs. 79.4%), MobileOne's latency is 2.8× slower than that of GhostNetV3 on CPU.

Table 9: Top-1 accuracy achieved by various compact models on the ImageNet dataset.

Method	Params (M)	FLOPs (M)	Latency(ms) CPU	Mobile	Top-1	Top-5
GhostNetV1 0.5× [10]	2.6	42	1.98	5.43	66.2	86.6
Mobileformer-52 [2]	3.6	52	34.27	89.65	68.7	—
MobileNetV3-S [15]	2.5	56	2.10	5.47	67.4	—
GhostNetV3 0.5× (ours)	2.7	48	2.09	5.67	**69.4**	**88.5**
Mobileformer-96 [2]	4.6	96	42.18	101.37	72.8	—
ShuffleNetV2 1.0× [18]	2.3	146	4.69	11.07	69.4	88.9
ShuffleNetV2 1.0× + ours	2.3	146	4.69	11.07	71.6	89.8
MobileViT-XXS [19]	1.3	373	12.5	29.47	69.0	—
MobileNetV1 [16]	4.2	575	7.02	12.85	70.6	—
MobileOne-S0 [28]	2.1	275	5.48	12.34	71.4	—
PiT-ti [13]	4.9	710	21.87	45.34	71.3	—
DeiT-tiny [27]	5.9	1300	28.01	56.4	72.2	—
PiT-xs [13]	10.6	1400	32.92	69.7	72.4	—
ConViT-tiny [9]	5.7	1000	28.75	58.73	73.1	91.7
MobileViT-XS [19]	2.3	724	23.43	64.34	74.8	—
MixNet-S [25]	4.1	256	15.75	30.79	75.8	92.8
MobileNetV3-L [15]	5.4	219	7.25	14.24	75.2	—
ShuffleNetV2 2.0× [18]	7.4	591	15.62	25.36	74.9	—
MNASNet-A1 [22]	3.9	312	7.81	13.57	75.2	92.5
MobileNetV2 1.0× [21]	3.4	300	7.96	14.97	72.0	90.8
MobileNetV2 1.0× + ours	3.4	300	7.96	14.97	75.0	92.1
GhostNetV1 1.0× [10]	5.2	141	6.25	12.65	73.9	91.4
MobileNeXt 1.0× [36]	3.4	311	8.63	16.17	74.0	—
MobileOne-S1 [28]	4.8	825	14.86	27.21	75.9	—
FBNet-C [30]	5.5	375	9.37	15.67	74.9	—
GhostNetV2 1.0× [10]	6.1	167	7.81	14.69	75.3	92.4
GhostNetV3 1.0× (ours)	6.1	167	7.81	14.69	**77.1**	**93.3**
MobileNetV2 1.4× [21]	6.9	585	13.07	22.85	74.7	—
GhostNetV1 1.3× [10]	7.3	226	12.69	24.31	75.7	93.4
MobileNeXt 1.4× [36]	6.1	590	11.46	21.65	76.1	—
MobileOne-S2 [28]	7.8	1299	23.04	40.28	77.4	—
MixNet-M [25]	5.0	360	21.44	43.12	77.0	93.3
MobileViT-S [19]	5.6	1792	35.93	92.28	78.4	—
GhostNetV2 1.3× [10]	8.9	269	14.46	28.69	76.9	93.4
GhostNetV3 1.3× (ours)	8.9	269	14.46	28.69	**79.1**	**94.5**
GhostNetV1 1.7× [10]	11.0	378	18.45	37.24	77.2	93.4
EfficientNet-B0 [23]	5.3	390	12.50	37.67	76.3	93.2
GhostNetV2 1.6× [26]	12.3	399	18.87	38.46	77.8	93.8
MobileOne-S3 [28]	10.1	1896	33.31	58.33	78.1	—
MixNet-L [25]	7.3	565	34.01	67.93	78.9	94.2
Mobileformer-508 [2]	14.0	508	129.58	163.39	79.3	—
MobileOne-S4 [28]	14.8	2978	52.86	92.06	79.4	—
GhostNetV3 1.6× (ours)	12.3	399	18.87	38.46	**80.4**	**95.2**

When comparing GhostNetV3 1.0× with search-based compact models, it outperforms FBNet-C citefbnet by 2.2% with faster inference speed on both CPU and phone. Additionally, GhostNetV3 1.0× provides a 1.9% top-1 accuracy advantage over MobileNetV3 and MNASNet, while maintaining similar latency. The results demonstrate the superiority of our proposed training strategy over existing manually designed and search-based architecture designing approaches in obtaining excellent compact models.

Figure 5 presents a comprehensive performance comparison of various compact models. The left and right figures illustrate the FLOPs and the latency measured on a mobile phone, respectively. Notably, our trained GhostNetV3 stands out by exhibiting the best balance between latency and top-1 accuracy on mobile devices.

Other compact models To further demonstrate the scalability of the proposed training strategy, we apply them to the training of two other widely used compact models: ShuffleNetV2 and MobileNetV2. The results in Table 9 show that our proposed strategy can improve the top-1 accuracy of ShuffleNetV2 and MobileNetV2 by 2.2% and 3.0%, respectively.

5.6   Extend to object detection

To investigate whether the training receipts would work well for other datasets, we expand our experiments to the object detection task on COCO to validate their generalization. The results are presented in Table 10. Notably, the insights from the classification task are applicable to the object detection task. For instance, the GhostNetV3 model outperforms GhostNetV2 by mAPs of 0.4 and 0.5 under the two used resolution settings, respectively. Additionally, GhostNetV3 outperforms MobileNetV2 while requires fewer FLOPs for inference.

Table 10: Performance of GhostNetV3 on object detection.

Backbone	Resolution	FLOPs(M)	mAP
MobileNetV2		613	22.2
GhostNetV1 1.1×	320×230	338	21.8
GhostNetV2 1.0×		342	22.3
GhostNetV3 1.0×		342	**22.7**
MobileNetV2		1035	23.9
GhostNetV1 1.1×	416×416	567	23.4
GhostNetV2 1.0×		571	24.1
GhostNetV3 1.0×		571	**24.6**

6   Conclusion

In this paper, we present a comprehensive study on training strategies aimed at enhancing the performance of existing compact models. The techniques, including re-parameterization, knowledge distillation, data augmentation, and learning schedule adjustments, involve no modifications to the model architecture during inference. In particular, our trained GhostNetV3 achieves an optimal balance between accuracy and inference costs, as verified on both CPU and mobile phone platforms. We also apply the proposed training strategy to other compact models such as MobileNetV2 and ShuffleNetV2, where significant improvements in accuracy are observed. We hope that our study can provide valuable insights and experiences for future research in this field.

References

1. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9640–9649 (2021) 5
2. Chen, Y., Dai, X., Chen, D., Liu, M., Dong, X., Yuan, L., Liu, Z.: Mobile-former: Bridging mobilenet and transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5270–5279 (2022) 4, 12, 13
3. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501 (2018) 7
4. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: 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) 7
5. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2009) 2, 8
6. Ding, X., Zhang, X., Han, J., Ding, G.: Diverse branch block: Building a convolution as an inception-like unit. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10886–10895 (2021) 2, 6
7. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: Repvgg: Making vgg-style convnets great again. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13733–13742 (2021) 2, 6
8. 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) 4
9. d'Ascoli, S., Touvron, H., Leavitt, M.L., Morcos, A.S., Biroli, G., Sagun, L.: Convit: Improving vision transformers with soft convolutional inductive biases. In: International Conference on Machine Learning. pp. 2286–2296. PMLR (2021) 13
10. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: Ghostnet: More features from cheap operations. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1580–1589 (2020) 1, 4, 13
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 2, 8, 9

12. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 558–567 (2019) 2, 5

13. Heo, B., Yun, S., Han, D., Chun, S., Choe, J., Oh, S.J.: Rethinking spatial dimensions of vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11936–11945 (2021) 13

14. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015) 3

15. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1314–1324 (2019) 1, 4, 12, 13

16. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) 1, 3, 12, 13

17. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016) 3

18. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV). pp. 116–131 (2018) 4, 12, 13

19. Mehta, S., Rastegari, M.: Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. arXiv preprint arXiv:2110.02178 (2021) 1, 4, 13

20. Peng, Z., Dong, L., Bao, H., Ye, Q., Wei, F.: Beit v2: Masked image modeling with vector-quantized visual tokenizers. arXiv preprint arXiv:2208.06366 (2022) 8, 9

21. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018) 1, 3, 12, 13

22. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2820–2828 (2019) 4, 12, 13

23. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International conference on machine learning. pp. 6105–6114. PMLR (2019) 12, 13

24. Tan, M., Le, Q.: Efficientnetv2: Smaller models and faster training. In: International conference on machine learning. pp. 10096–10106. PMLR (2021) 2, 5

25. Tan, M., Le, Q.V.: Mixconv: Mixed depthwise convolutional kernels. arXiv preprint arXiv:1907.09595 (2019) 12, 13

26. Tang, Y., Han, K., Guo, J., Xu, C., Xu, C., Wang, Y.: Ghostnetv2: Enhance cheap operation with long-range attention. arXiv preprint arXiv:2211.12905 (2022) 1, 2, 4, 13

27. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021) 2, 8, 9, 13

28. Vasu, P.K.A., Gabriel, J., Zhu, J., Tuzel, O., Ranjan, A.: An improved one millisecond mobile backbone. arXiv preprint arXiv:2206.04040 (2022) 1, 12, 13

29. Wightman, R., Touvron, H., Jégou, H.: Resnet strikes back: An improved training procedure in timm. arXiv preprint arXiv:2110.00476 (2021) 2, 5

30. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: 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) 12, 13
31. You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., Hsieh, C.J.: Large batch optimization for deep learning: Training bert in 76 minutes. arXiv preprint arXiv:1904.00962 (2019) 7
32. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6023–6032 (2019) 7
33. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017) 7
34. Zhang, X., Zhou, X., Lin, M., Sun, J.: 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) 4, 12
35. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 13001–13008 (2020) 7
36. Zhou, D., Hou, Q., Chen, Y., Feng, J., Yan, S.: Rethinking bottleneck structure for efficient mobile network design. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. pp. 680–697. Springer (2020) 4, 12, 13