This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

RepAn: Enhanced Annealing through Re-parameterization

Xiang Fei[1,2†], Xiawu Zheng[1,2,3†], Yan Wang[4], Fei Chao[1,2], Chenglin Wu[5], Liujuan Cao[1,2*]
[1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing.
Ministry of Education of China, Xiamen University. 361005, PR. China.
[2]School of Informatics, Xiamen University, China. [3]Peng Cheng Laboratory.
[4]Samsara, 1 De Haro Street, San Francisco, California, USA. [5]DeepWisdom Inc.
{xiangf, zhengxiawu}@stu.xmu.edu.cn, yan.wang@samsara.com
fchao@xmu.edu.cn, alexanderwu@deepwisdom.ai, caoliujuan@xmu.edu.cn

Abstract

The simulated annealing algorithm aims to improve model convergence through multiple restarts of training. However, existing annealing algorithms overlook the correlation between different cycles, neglecting the potential for incremental learning. We contend that a fixed network structure prevents the model from recognizing distinct features at different training stages. To this end, we propose RepAn, redesigning the irreversible re-parameterization (Rep) method and integrating it with annealing to enhance training. Specifically, the network goes through Rep, expansion, restoration, and backpropagation operations during training, and iterating through these processes in each annealing round. Such a method exhibits good generalization and is easy to apply, and we provide theoretical explanations for its effectiveness. Experiments demonstrate that our method improves baseline performance by 6.38% *on the CIFAR-100 dataset and* 2.80% *on ImageNet, achieving state-of-the-art performance in the Rep field. The code is available at https://github.com/xfey/RepAn.*

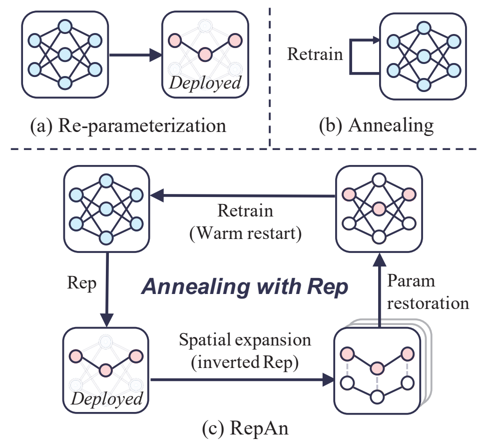

Figure 1. **(a)** The Rep approach is irreversible. **(b)** Annealing simply restarts training cyclically. **(c)** Our approach investigates the efficacy of Rep in enhancing model accuracy with annealing.

1. Introduction

Convolutional neural networks (CNNs) [39] have achieved remarkable results in the field of computer vision [27, 30, 36, 43, 52]. Among them, some classical network architectures such as VGG [56], ResNet [26], DenseNet [34] and MobileNet [31] have achieved great success by stacking convolutional modules. Practically, one needs to consider the trade-off of the model's overall performance, including accuracy, inference speed, and memory footprint.

The Re-parameterization technique (Rep) [17, 65] is inspired by the characteristics of neural architecture and aims to achieve cost-free efficiency improvements. Rep involves a series of branch fusion operations that merge multiple parallel branches into a single layer. During training, the network utilizes a multi-branch structure to aid in optimization. After convergence, for convolutional and batch normalization (BN) [37] layers, lossless merging operations can be performed to achieve accelerated inference.

Despite the effectiveness of Rep, its application to model training is not only counter-intuitive but also technically challenging. The branch merging operation, designed for model deployment, is practically a one-way procedure with an ill-posed inverse operator. As a result, research on Rep is primarily focused on the diversity of its compatible structures [14, 16, 58]. In this paper, we explore the overlooked potential of Rep to benefit the training accuracy of networks. Figure 1 illustrates the difference between the traditional application of Rep and our proposed method.

*Corresponding author.
†These authors contributed equally to this work.

The simulated annealing algorithms [45, 57] have been employed to cyclically restart the training of the network, re-initializing the network for several rounds, aiming to achieve better convergence. However, existing algorithms neglect the changes within the model, relying solely on a fixed structure for repetition. These methods ignore the continuity between different training stages, resulting in the loss of temporal knowledge inheritance [33]. In this paper, we introduce structural changes into the annealing process, to achieve the effect of incremental learning.

Our proposed method, RepAn, involves incorporating the core operation of branch merging from Rep into the annealing training process. The primary challenge lies in transforming the irreversible Rep into a recyclable workflow, and integrating it effectively into the annealing algorithm. To address this issue, we devised two additional stages, namely structural expansion and parameter restoration. The workflow is depicted in Fig. 1(c), and consists of three simple steps: (i) *Re-parameterization.* Merging the multi-branch network into a single-branch structure via Rep. (ii) *Unfolding.* The model is recovered to a trainable one through expansion and restoration operations. (iii) *Training.* Each annealing training cycle replicates the aforementioned operations, enhancing the training effectiveness. The overview of our method is shown in Fig. 2, and the details of RepAn is described in Sec. 3.

We also present a possible explanation for the effectiveness of our work in Sec. 4. In each cycle, Rep is applied to inherit previously learned knowledge through lossless compression. This allows the network to preserve its performance while reducing its memory footprint. Subsequently, new branches are introduced and trained to learn additional features of the current cycle. Adhering to this cyclic procedure leads to an effective annealing training, thus adopts an incremental learning and ensemble approach, as evidenced by our experimental results.

Our approach is compatible with all Rep architectures, making it a flexible and well-generalized training paradigm. By integrating knowledge and continuously enhancing the network's capability, our method achieves improvements in performance, leading to a accuracy gain of 6.38% on the CIFAR-100 dataset [38] and 2.80% on ImageNet [13].

Our contributions are summarized as follows.

- For the first time, we explore using Rep for accuracy enhancement, taking advantage of its lossless compression property and designing a new cyclic annealing training workflow termed RepAn.
- We present a theoretical explanation and proof of the effectiveness of our method, enhancing the fitting capability with better optimization procedure.
- Extensive experiments on various datasets, structures, and downstream tasks verify that our method improves the performance without increasing extra parameters.

2. Structural Re-parameterization

This section introduces basic definitions and preliminaries to derive the principles of the re-parameterization, and the related applications.

2.1. Problem Formulation and Preliminaries

For a convolutional layer F with C_{in} input channels, C_{out} output channels and a kernel size of K, parameters of F are denoted as $\boldsymbol{W} \in \mathbb{R}^{C_{out}\times C_{in}\times K\times K}$, with an optional bias term, $\boldsymbol{b} \in \mathbb{R}^{C_{out}}$. For an input feature map $\boldsymbol{X} \in \mathbb{R}^{C_{in}\times H\times W}$, its forward propagation through a convolutional layer is formulated as: $\mathrm{F}\left(\boldsymbol{X}\right) = \boldsymbol{X} \circledast \boldsymbol{W} + \boldsymbol{b}$.

Convolution Linearity. For convolutional layers with the same configurations (*e.g.*, filter size, stride, input and output channels), the linearity of convolutional operations follows the additive constancy:

$$\begin{aligned}\mathrm{F}_1\left(\boldsymbol{X}\right)+\mathrm{F}_2\left(\boldsymbol{X}\right) &= \left(\boldsymbol{X}\circledast\boldsymbol{W}_1+\boldsymbol{b_1}\right)+\left(\boldsymbol{X}\circledast\boldsymbol{W}_2+\boldsymbol{b_2}\right)\\ &= \boldsymbol{X}\circledast\left(\boldsymbol{W}_1+\boldsymbol{W}_2\right)+\left(\boldsymbol{b_1}+\boldsymbol{b_2}\right).\end{aligned} \tag{1}$$

Let $\boldsymbol{W'} = \boldsymbol{W}_1 + \boldsymbol{W}_2$ and $\boldsymbol{b'} = \boldsymbol{b_1} + \boldsymbol{b_2}$, then the constructed convolution satisfies $\mathrm{F}'\left(\boldsymbol{X}\right) = \mathrm{F}_1\left(\boldsymbol{X}\right) + \mathrm{F}_2\left(\boldsymbol{X}\right)$.

BN Fusion and Branch Integration. The Batch Normalization (BN) [37] layer can be fused into its preceding convolutional layer while retaining the output unchanged. The BN-Conv module is formulated as:

$$\begin{aligned}\mathrm{BN}\left(\mathrm{F}\left(\boldsymbol{X}\right)\right) &= \frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}}\left(\boldsymbol{X}\circledast\boldsymbol{W}+\boldsymbol{b}-\boldsymbol{\mu}\right)+\boldsymbol{\beta}\\ &= \boldsymbol{X}\circledast\left(\frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}}\boldsymbol{W}\right)+\left[\frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}}\left(\boldsymbol{b}-\boldsymbol{\mu}\right)+\boldsymbol{\beta}\right].\end{aligned} \tag{2}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the accumulated mean and standard deviation of the BN layer, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ denote the learned scaling factor and the bias term, respectively. Let $\boldsymbol{W'} = \frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}}\boldsymbol{W}$ and $\boldsymbol{b'} = \frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}}\left(\boldsymbol{b}-\boldsymbol{\mu}\right)+\boldsymbol{\beta}$, then the constructed convolutional layer satisfies $\mathrm{F}'\left(\boldsymbol{X}\right) = \mathrm{BN}\left(\mathrm{F}\left(\boldsymbol{X}\right)\right)$.

According to the above fusion methods, the Rep network can absorb multiple Conv-BN modules from parallel branches into a single convolutional layer.

2.2. Related Work

The re-parameterization (Rep) technique [17] is originally proposed to accelerate the inference time of neural networks. During training, each block contains multiple parallel branches; when the model converges, parallel branches are merged into a mathematically equivalent convolutional layer by following Eqs. (1) and (2). With the help of Rep, the RepVGG network achieved a speedup of 83% compared to ResNet-50 [17, 26]. Existing methods have primarily focused on the compatibility with advanced architectures to gain better performance, such as asymmetric convolution [40], average pooling [16] and residual connection [26]. They increase the network's capacity during training, and uses Rep processes to accelerate inference.

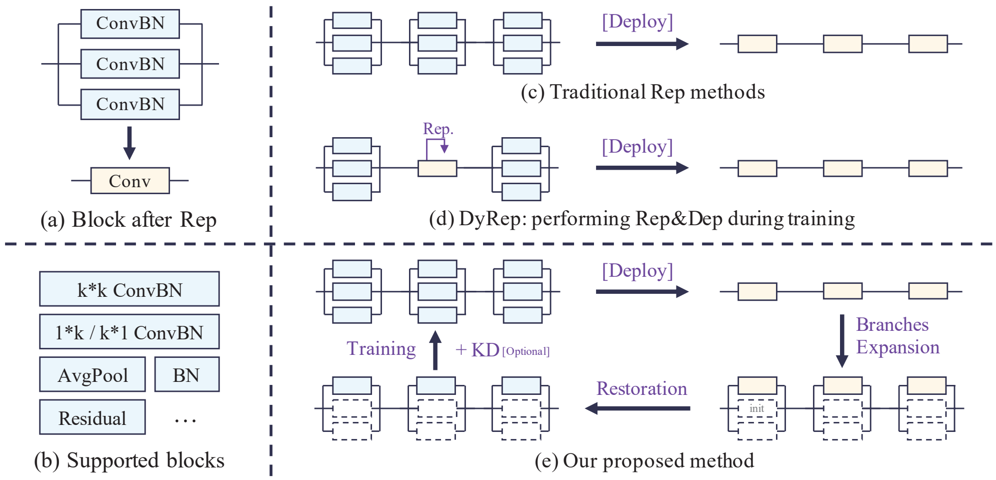

Figure 2. Overview of the proposed RepAn. **(a)** The Re-parameterization operation. **(b)** Supported blocks by Rep methods. **(c)** Rep is irreversibly used during deployment only by traditional methods. **(d)** DyRep [35] dynamically performs Rep&Dep during training to reduce memory consumption. **(e)** RepAn adheres to a cyclic training flow, inheriting knowledge and facilitating incremental learning to enhance the effectiveness of annealing training.

Rep-based modules are also widely used in other fields, involving various hybrid architectures. Transformer-based structures [21, 48, 59] showed excellent performance recently, and RepMLPNet [18] helps the backbones better exploit local attentions. RepAdapter [46] improves efficiency of vision-language pretrain (VLP) tasks. Neural architecture search [25, 68, 69] aims to find suitable structures under constraints, and RepNAS [66] adaptively adds branches under heavier payloads. Rep can also be regarded as a structural compression [20, 47] method, thus ResRep [15] applies Rep into network pruning. Several innovative models also use Rep to accelerate their models, *e.g.*, large convolution kernels are used in RepLKNet [19]. MobileOne [58] and YOLOv7 [60] follows Rep rules to build backbones and achieves low latency on mobile devices.

Few works have discussed the role of Rep during training, and we draw inspiration from the DyRep [35]. It dynamically adjusts network layers during different epochs of training, aiming to reduce the training FLOPs. While it focuses on memory savings achieved by *compressing*, our approach, conversely, aims at the performance gains resulting from the *expansion*. We also illustrate the differences in Fig. 2. The parameter reinitialization [2] has also provided us with insights. This approach cyclically resets a portion of the parameters and then re-trains the network. We aim to continually benefit network training while minimizing disruption to the parameters.

The Rep technique has been demonstrated to be effective on various neural networks and tasks. However, most existing methods treat Rep merely as a deployment acceleration technique. In contrast, our study aims to explore the potential of Rep for accuracy enhancement.

3. Annealing through Rep

In this section, we start with the explanation of using Rep as a lossless compression technique in Sec. 3.1, and then describe the specific process of our proposed method, which is divided into three stages: branches expansion (Sec. 3.2), BN restoration (Sec. 3.3) and training (Sec. 3.4). After completing the training, the network is compressed again using Rep, recursively following the annealing process. Finally, we give a theoretical explanation of the feasibility in Sec. 4.

3.1. Rep as Lossless Compression

The Rep technique is primarily used to accelerate inference and reduce computational cost. However, we believe that Rep can also improve model accuracy. This raises the question: *why has Rep not been used in this manner before?*

In order to address this issue, we first examine the implementation of Rep. We observe that the branch merging operation, which is a key component of Rep, is essentially irreversible as it incorporates the Batch Normalization (BN) parameters into the convolutional layers through numerical multiplication of weights. This makes it difficult to normalize gradients, rendering the transformed network unsuitable for fine-tuning and the Rep process irreversible. Therefore, preserving the original BN structure is crucial for ensuring the network can be continuously trained.

To implement the RepAn algorithm, we first need to expand the branches and restore the BN structures. In order to achieve this, we have designed corresponding methods which are described in detail in Secs. 3.2 and 3.3. We also have found that knowledge distillation can effectively enhance the training process during structural modification. This can be viewed as an implementation of both the ensemble [4, 5, 54] and incremental learning [1, 50, 51] approaches. While traditional methods preserve newly learned knowledge through additional structures, our approach takes full advantage of re-parameterization without changing the network architecture.

We integrate Rep into the annealing algorithm, proposing RepAn, which comprises the following three steps: (i) *Re-parameterization.* The multi-branch network is losslessly merged into a single-branch one using Rep, which is used for deployment by traditional methods. (ii) *Re-expand.* By adding parallel branches to the compressed model and learning additional knowledge, the network capability is improved. (iii) *Recursive.* If the multi-branch structure is the same as the network, this process can be recursively performed to further enhance the network's performance.

The overall workflow of RepAn is illustrated in Fig. 2. Specifically, the annealing process goes through cycles of re-parameterization, expansion, restoration, and backpropagation operations. The implementation and theoretical explanation are specified as follows.

3.2. Branches Expansion

To facilitate continuous training, the network structure is rejuvenated through the addition of randomly initialized parallel branches. Each convolutional layer is expanded into a re-parameterization block, following the computational rules outlined in Eqs. (1) and (2). Diverse structures of branches have been proposed in prior research [14, 16, 17], shown in Fig. 2(b).

Without losing generality, we design an adjustment scheme $attach\_rate$ for these branches to aid optimizations, denoted by the symbol λ and $0 \leq \lambda \leq 1$. The output $\boldsymbol{Y}$ of the expanded block is represented as

$$\begin{aligned} \boldsymbol{Y} &= \boldsymbol{Y}_{\text{inv}} + \lambda \left(\boldsymbol{Y}_{\text{exp}} + \boldsymbol{Y}_{\text{res}} \right) \\ &= \text{BN}_{\text{inv}} \left(\text{F}_{\text{inv}} \left(\boldsymbol{X} \right) \right) \\ &+ \lambda \left[\sum_i \text{BN}_{\text{exp}}^{(\text{i})} \left(\text{F}_{\text{exp}}{}^{(i)} \left(\boldsymbol{X} \right) \right) + \text{BN}_{\text{res}} \left(\boldsymbol{X} \right) \right], \end{aligned} \tag{3}$$

where the subscripts represent the inverted (inv), the expanded (exp) and the residual connection (res) branches, respectively. Considering that multiple branches work in parallel, the summation symbol is used to combine the output of all the branches, based on the linearity of convolution as demonstrated in Eq. (1). The value of λ controls the training process, where setting $\lambda = 0$ ensures that Eq. (3) maintains the network's previous output. As λ increases, the contributions of the expanded branches become more significant, facilitating the network's ability to learn new knowledge.

3.3. BN Restoration

The absence of a normalization layer in a pure convolutional network makes normal training difficult. Hence, after branch expansion, restoring the BN structure is necessary. However, we noticed that directly performing training with randomly initialized weights may cause instability during early stages of training, which can significantly affect inherited branch weights. We observed that BN layer initialization at the start of training can cause convolutional layer weights to change. To facilitate the subsequent training process, we propose an independent BN recovery stage.

In contrast to the fusion procedure, since the convolutional layer here has been re-parametrically processed (which can be abbreviated as Rep-conv) to contain the affine transform of BN operations, it is necessary to invert Eq. (2) to construct the convolutional layer with the BN layer, then we have:

$$\begin{aligned} \text{BN} \left(\text{F}' \left(\boldsymbol{X} \right) \right) &= \frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}} \left(\boldsymbol{X} \circledast \boldsymbol{W}' + \boldsymbol{b}' - \boldsymbol{\mu} \right) + \boldsymbol{\beta} \\ &= \boldsymbol{X} \circledast \left(\frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}} \boldsymbol{W}' \right) + \left[\frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}} \left(\boldsymbol{b}' - \boldsymbol{\mu} \right) + \boldsymbol{\beta} \right]. \end{aligned} \tag{4}$$

For the given BN parameters, let $\boldsymbol{W}' = \frac{\boldsymbol{\sigma}}{\boldsymbol{\gamma}} \boldsymbol{W}$ and $\boldsymbol{b}' = \frac{\boldsymbol{\sigma}}{\boldsymbol{\gamma}} \left(\boldsymbol{b} - \boldsymbol{\beta} \right) + \boldsymbol{\mu}$, and the constructed convolution satisfies the inverted version of Eq. (2). This step restores the BN layers to help subsequent training proceed smoothly.

During the recursive procedure, the BN layer parameters can be inherited directly from the previous step. Alternatively, the parameters can also be computed during the forward propagation process using calibration methods [6, 35, 61, 64]. Calibration uses a single batch of data to stabilize the weight values. As the branches are expanded, a data batch is used to perform forward propagation through the Conv-BN blocks, and the BN coefficients are adjusted to maintain stable mean and variance. The parameters of both the convolutional and BN layers can be iteratively updated, which also mitigates the impact of additional initialization branches. This effect will be further analyzed in Sec. 5.2.

3.4. Learning Strategies

Optionally, a knowledge-oriented learning strategy enhances the training performance of RepAn, *e.g.*, the incorporation of Knowledge Distillation (KD) [23, 28]. KD uses a high-performing teacher network to guide the student network, with soft labels for more accurate training. The annealing training strategy enhances KD's ability to inherit knowledge, contributing to RepAn achieving state-of-the-art performances. Our experiments in Sec. 5.1 validate this claim. In addition, other training techniques such as bootstrap [44], hard example mining [55], and curriculum learning [3, 24] can also be used for progressive training.

The training process for RepAn is presented in Algorithm 1. In summary, our work highlights the overlooked

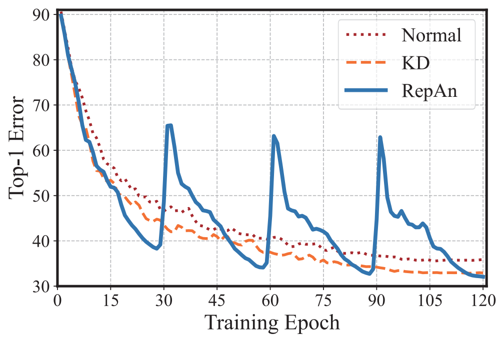

Figure 3. The learning curve comparison between RepAn and traditional training methods.

Algorithm 1: Training with RepAn

Input: Rep network $\mathcal{N}$ with weights $\boldsymbol{w}$, Teacher network $\mathcal{N}_T$, Train dataset $\mathcal{D}$, Adjustment scheduler Θ, Number of cycles R, Number of training epochs E.

Output: Network $\mathcal{N}$ with optimal weights $\boldsymbol{w}$

Initialize $\mathcal{N}$ or load pretrained weights for $\boldsymbol{w}$

Switch $\mathcal{N}$ to deployment: $\mathcal{N}_{deploy} \leftarrow \text{Rep}\,(\mathcal{N})$

for $r = 1, \ldots, R$ **do**

  Expand new branches: $\mathcal{N} \leftarrow \text{Expand}\,(\mathcal{N}_{\text{deploy}})$

  Restoration: $\boldsymbol{w}_{train} \leftarrow \text{Restore}\,(\boldsymbol{w}_{\text{deploy}})$

  Calibrate BN: $\boldsymbol{w} \leftarrow \text{Calibrate}\,(\boldsymbol{w}_{train})$

  for $e = 1, \ldots, E$ **do**

    Update λ in Eq. (3): $\lambda \leftarrow \Theta\,(r, e)$

    Network training: $\text{Train}\,(\mathcal{N}, \mathcal{D}, \mathcal{N}_T)$

  end

  Switch to deployment: $\mathcal{N}_{deploy} \leftarrow \text{Rep}\,(\mathcal{N}_{\text{train}})$

end

potential of Rep to benefit training, which is an evolution of conventional Rep methods and is compatible with all re-parameterized structures. It can be regarded as incremental learning with lossless compression, improving the network during the annealing procedure.

4. Why RepAn Works

The proposed method leverages Rep to better activate the performance of the annealing algorithm. We demonstrate the efficacy of our method through the lens of ensemble learning and training procedures, and conduct experiments to validate our claims.

Ensemble and Incremental Learning. Ensemble learning techniques [4, 5] aggregate the outputs of multiple models to obtain better predictions. We are inspired by these methods during training [7, 33, 62] and find that models at different epochs can also be integrated, similar to the Exponential Moving Average (EMA) technique [29].

Our proposed method, RepAn, achieves a similar effect to the Snapshot ensemble [33]. By inheriting the weights and training iteratively, our method converges to a better endpoint as the model is progressively compressed losslessly and applied to the next round by Rep. As a result, knowledge from different stages is integrated, which enables implicit ensemble learning. As shown in Fig. 3, the model starts from a better initialization after each round and eventually converges to a better endpoint.

Additionally, our method avoids learning from repeated representations by performing forward propagation together with the inherited branches. This characteristic aligns with the definition of incremental learning [9, 49, 50], where different parameters are used to learn additional knowledge. Since the Rep process merges different branches into a single convolutional layer, our method extends the branches to gain additional representation capability.

The proposed training process of RepAn has several benefits: it retains learned knowledge, focusing on new branches for faster optimization and reducing required training epochs. Inheriting knowledge also accelerates new branch learning and improves convergence.

We have designed a toy example using RepVGG [17] on the CIFAR-100 dataset to verify the effectiveness of the RepAn approach in implementing the aforementioned learning methods. The results of the experiment are shown in Fig. 4. The use of multiple branches provides a higher capacity than a single branch, resulting in better final performance. In this example, we loaded the weights from the single branch into the multi-branch structure for initialization. However, due to the modification of the branch structures, the parameters of the BN layers needed to be updated, resulting in a significantly larger loss value at the beginning of the training. Nonetheless, the training loss value decreased rapidly in the early epochs, and the network eventually converged to a better result. This example demonstrates that

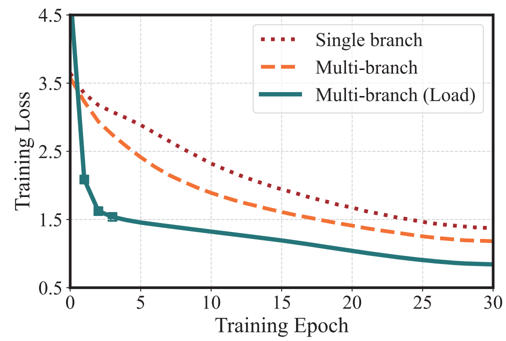

Figure 4. Toy example of learning different branches. We compare single-branch, multi-branch, and multi-branch structures that inherit the weights from the single one.

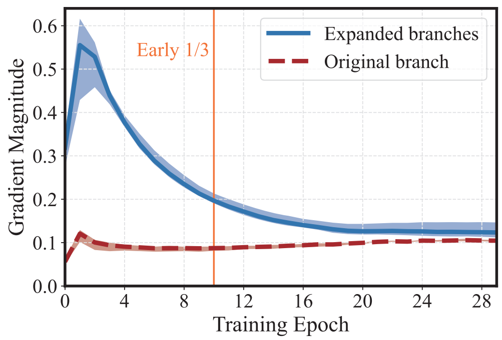

Figure 5. Gradient magnitude comparison between the original branch and expanded branches.

Network	Method	Top-1 Accuracy C-10	Top-1 Accuracy C-100	Accuracy ↑ C-10	Accuracy ↑ C-100
RepVGG (A1) [17]	Baseline	89.60	64.90	–	–
	+KD [28]	91.53	67.24	1.93	2.34
	Ours	**92.01**	**71.28**	**2.41**	**6.38**
RepVGG (B1)	Baseline	92.39	68.97	–	–
	+KD	93.32	72.18	0.93	3.21
	Ours	**93.98**	**74.57**	**1.59**	**5.60**
RepVGG (B3)	Baseline	92.84	71.71	–	–
	+KD	93.49	75.11	0.65	3.40
	Ours	**94.32**	**76.89**	**1.48**	**5.18**
ResNet-18 (DBB) [16, 26]	Baseline	93.74	73.44	–	–
	+KD	94.19	76.96	0.45	3.52
	Ours	**94.68**	**77.97**	**0.94**	**4.53**
ResNet-18 (ACNet) [14]	Baseline	93.96	73.90	–	–
	+KD	93.89	77.94	-0.07	4.04
	Ours	**94.44**	**78.37**	**0.48**	**4.47**

Table 1. Top-1 accuracy of five networks using different training methods on the CIFAR-10/100 [38] datasets.

similar approach like ours can achieve incremental learning by inheriting the weights, and that knowledge from earlier stages is integrated into the network. In addition, the necessity of recovering BN parameters is also proved.

Gradient Analysis of Expanded Modules. Analyzing the differences in gradients can provide insight into whether knowledge is being transferred correctly. Combining the convolutional linearity mentioned in Eq. (1) and the fusion methods mentioned in Eq. (2), the convolution parameters of Eq. (3) can be expanded as follows:

$$\begin{aligned} \boldsymbol{Y} = \boldsymbol{X} \circledast \left(\frac{\gamma_{\text{inv}}}{\boldsymbol{\sigma}_{\text{inv}}} \boldsymbol{W}_{\text{inv}} + \lambda \sum_i \frac{\gamma_{\text{exp}}^{(i)}}{\boldsymbol{\sigma}_{\text{exp}}^{(i)}} \boldsymbol{W}_{\text{exp}}^{(i)} \right) \\ + \frac{\gamma_{\text{inv}}}{\boldsymbol{\sigma}_{\text{inv}}} \boldsymbol{b}_{\text{inv}} + \lambda \sum_i \frac{\gamma_{\text{exp}}^{(i)}}{\boldsymbol{\sigma}_{\text{exp}}^{(i)}} \boldsymbol{b}_{\text{exp}}^{(i)} + C, \end{aligned} \tag{5}$$

where C is the other term that is not related to the convolution parameters $\boldsymbol{W}$ and $\boldsymbol{b}$. According to Eq. (5), the gradient on the convolution parameters of the expanded branches can be calculated as:

$$\begin{cases} \frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{W}_{\text{exp}}} = \boldsymbol{X} \circledast \lambda \sum_i \left(\frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{Y}} \cdot \frac{\gamma_{\text{exp}}^{(i)}}{\sigma_{\text{exp}}^{(i)}} \right), \\ \frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{b}_{\text{exp}}} = \lambda \sum_i \left[\sum_{u,v} \left(\frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{Y}} \right)_{u,v} \cdot \frac{\gamma_{\text{exp}}^{(i)}}{\sigma_{\text{exp}}^{(i)}} \right], \end{cases} \tag{6}$$

where the parameters u and v are the expansions of the two dimensions of the value, which are then summed to match the shape of the bias term $\boldsymbol{b}$.

The newly added branches will inevitably perturb the original output, causing a risk of oscillation in the gradient changes. To reduce such adverse effects, DyRep [35] proposes to modify the BN parameter γ artificially during training. Since γ is updated during the forward propagation, we argue that such manual intervention could potentially interfere with gradient transfer. However, the introduced parameter λ works equivalently to the $\sum_i \frac{\gamma_{\text{exp}}^{(i)}}{\sigma_{\text{exp}}^{(i)}}$ in Eq. (6), and can also be scheduled manually. Therefore, adjusting λ is more efficient and convenient for facilitating training.

We also compared the mean of the absolute gradient value of different branches, as shown in Fig. 5. In the early stages, the gradient magnitude of the expanded branches is several times higher than the original branch's. This also allows for better retention of inherited knowledge in the early stages of training. This is due to the fact that the original branch has already been well-trained and is less affected by the perturbations caused by the addition of new branches, whereas the newly expanded components are not. This characteristic contributes to better retention of inherited knowledge in the early stages of training.

5. Experimentation

This section presents an evaluation of the proposed method on several widely-used datasets, structures and downstream tasks. Besides, ablation studies are performed to analyze the critical configurations of our method.

5.1. Comparison

We conducted initial experiments on the CIFAR-10 and CIFAR-100 datasets to validate the effectiveness of our proposed method. We used the VGG [56] and ResNet [26] architectures and compared them with re-parameterized branch structures from RepVGG [17], DBB [16], and ACNet [14]. To examine the training methods, we compared training with knowledge distillation (KD) [28]. This comparison is conducted because KD has the potential to enhance the effectiveness of this method, necessitating control experiments for elimination. In our experiments, we employ commonly-used soft labels for knowledge distillation training.

Method	Network			
	MobileNet	ResNet-18	ResNet-34	ResNet-50
Baseline	71.89	69.54	74.17	76.31
ACNet [14]	72.14	70.53	74.30*	76.46
DBB [16]	72.88	70.99	74.33	76.71
DyRep [35]	72.96	71.58	74.68	77.08
KD	73.16	71.88	75.64	77.11
Ours w/o KD	72.46	71.51	74.89	76.82
Ours	**73.43**	**72.34**	**76.50**	**77.76**

Table 2. ImageNet [13] top-1 accuracy of different training methods on MobileNet [31] and ResNet [26]. *: Our implementation.

Network	Speed	FLOPs (G)	Params (M)	Accuracy (%)
RepVGG-A1 [17]	1621	2.4	12.78	74.46
OREPA-A1 [32]	1621	2.4	12.78	74.85
ODBB-A1 [66]	1621	2.4	12.78	75.24
ResNet-34 [26]	1419	3.7	21.78	74.17
DBB-r34 [16]	1419	3.7	21.78	74.33
OREPA-r34	1419	3.7	21.78	75.04
KD-r34	1419	3.7	21.78	75.64
Ours-r34	1419	3.7	21.78	**76.50**
RepVGG-A2	1322	5.1	25.49	76.48
OREPA-A2	1322	5.1	25.49	76.72
ODBB-A2	1322	5.1	25.49	76.86
DyRep-A2 [35]	1322	5.1	25.49	76.91
ResNet-50	719	3.9	25.53	76.31
DBB-r50	719	3.9	25.53	76.71
KD-r50	719	3.9	25.53	77.11
Ours-r50	719	3.9	25.53	**77.76**
ResNeXt-50 [63]	484	4.2	24.99	77.46
ResNet-101	430	7.6	44.49	77.21
VGG-16 [56]	415	15.5	138.35	72.21
RepVGG-B3	363	26.2	110.96	80.52
ODBB-B3	363	26.2	110.96	80.97
DyRep-B3	363	26.2	110.96	81.12
ResNeXt-101	295	8.0	44.10	78.42
KD-B3	363	26.2	110.96	81.26
Ours-B3	363	26.2	110.96	**81.60**

Table 3. ImageNet top-1 accuracy of different Rep methods and baselines. The FLOPs and number of parameters are recorded during inference.

For these methods, we equivalently use a batch size of 128, a learning rate initialized to 0.2 and cosine annealing for 160 epochs. SGD [53] with weight decay of 10^{-4} is applied. For the RepVGG models, we use a reduced width of $0.25\times$ channels. Two additional parameters $\alpha = 0.9$ and temperature $= 10$ are used for the KD criterion. As illustrated in Fig. 3, RepAn uses fewer epochs and trains through multiple recursive steps. We set the epoch number to 30 and train for 5 time steps, making the maximum epochs $30 \times 5 = 150$ close to the baseline 160.

Table 1 presents the experimental results, which indicate that RepAn can improve the accuracy of the CIFAR-100 dataset by up to 6.38% and CIFAR-10 by up to 2.41%. Our method's generalizability is also verified by its performance on different Rep structures, achieving 4.53% and 4.47% accuracy improvements with DBB and ACNet on the CIFAR-100 dataset, respectively. RepAn shows remarkable performance improvement in all five settings. Such improvement is attributed to the effectiveness of the proposed training paradigm, which further confirms the method's effectiveness during the sanity check.

Training Method	KD Hyperparameters		Accuracy(%)	
	α	temperature	C-10	C-100
Baseline	–	–	89.60	64.90
RepAn	0	–	90.41	66.22
	0.5	4	91.20	69.53
	0.9	**10**	**92.01**	**71.28**
	0.9	20	91.57	70.40
	1.0	10	91.79	69.94

Table 4. Comparison of hyperparameters for knowledge distillation. The α represents the proportion of the KD criterion, and the temperature represents the softening effect on the label.

Network	Number of Parallel Branches				
	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
MobileOne-S0	70.9	70.7	71.3	71.4	71.1
MobileOne-S1	75.9	75.7	75.6	75.6	75.2
RepVGG-A1	64.9	66.3	67.4	68.0	68.3
RepVGG-A1 (KD)	67.2	68.7	69.1	69.9	70.5
RepVGG-A1 (Ours)	**71.3**	**72.5**	**72.8**	**73.3**	**74.2**

Table 5. Comparison of top-1 accuracy for various values k of parallel branches. The results for MobileOne [58] and RepVGG are obtained on the ImageNet and CIFAR-100 datasets, respectively.

Backbone	Method	ImageNet (top-1)	COCO (mAP) [41]	Cityscapes (mIOU) [11]
ResNet-18	Original	71.2	31.7	74.9
	Ours	**72.3**	**32.2**	**75.5**
ResNet-50	Original	76.3	36.3	77.8
	Ours	**77.8**	**36.7**	**78.2**

Table 6. Results on object detection and semantic segmentation tasks. Rep methods are adopted during ImageNet training.

We then perform validation on the ImageNet-1K [13] dataset, which contains 1.28M training images and 50K validation data in 1000 categories. We set the batch size to 256 on 8 GPUs, and train networks for 120 epochs with an additional 5 epochs to warm up. We apply an SGD optimizer and the cosine annealing scheduler, with an initial learning rate of 0.1. The KD hyperparameter α is reduced to 0.5 for larger datasets. For the RepVGG [17] models, we follow the configuration of the original implementation for both training and evaluation. We train for 3 time steps and report the final performance.

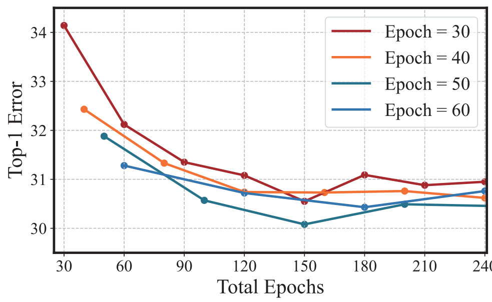

Figure 6. Comparison of using different training epochs for each recursive time step on CIFAR-100. Best viewed in color.

Table 2 shows comparison results of different training methods on ImageNet, and RepAn can bring up to 2.80% performance improvement to the baselines. On the RepVGG-B3 network, our method achieves a performance of 81.60%, reaching the state-of-the-art among Rep methods, as shown in Table 3.

5.2. Ablation Studies

Influence of Knowledge Distillation. Knowledge Distillation (KD) is optional for RepAn to facilitate training. As indicated in Tables 1 to 3, KD could slightly enhance the performance of conventional networks. However, our proposed method leverages the benefits of KD and achieves additional improvements in the experiments.

Training without KD. In addition to the preceding experiment, we conducted tests on training without KD. Specifically, we removed the teacher network by setting the distillation hyperparameter α to 0. Subsequently, we gradually increased the values to compare how different distillation ratios affect the training performance. The results in Tables 2 and 4 indicate that our method can also achieve performance improvement without KD. Furthermore, large KD hyperparameters may decrease the training performance due to the excessive modification of the training labels.

Influence of Parallel Branches. In this experiment, we explore the effect of diminishing marginal utility on multiple parallel branches in our approach. The results in Table 5 demonstrate that the performance improves slightly with the inclusion of more branches, but the improvement diminishes gradually. A study on MobileOne [58] suggests that parallel branches do not enhance the performance on ImageNet, which could be due to the networks having sufficient capacity. When using additional branches, accuracy improves by 2.9% compared to our best reported value. This observation suggests that there is still potential for further improvement in our method.

Influence of Epoch Number. As shown in Fig. 3, our method is trained cyclically with fewer epochs during each cycle. We recorded the final performance with different choices of epoch numbers in Fig. 6. Increasing the number of epochs during initial convergence can enhance model performance, but all models eventually reach optimal performance with sufficient epochs. Longer training improves data fitting, while shorter training times can better utilize recursive learning under similar constraints. Furthermore, overfitting can occur with too many training epochs, causing a slight decrease in performance.

Generalizability on Downstream Tasks. Our pre-trained models are applied as backbones for object detection task on the MS-COCO dataset using RetinaNet [42] and semantic segmentation task using PSPNet [67]. We use MMDetection [8] and MMSegmentation [10] with default settings to train these models. Our method outperforms the baselines at these tasks, as shown in Table 6.

More Ablation Studies. Comparison of schedulers for λ, results of different training epochs on ImageNet, and additional analyses are reported in supplementary material.

5.3. Discussion

On smaller datasets like CIFAR-10/100, our method outperforms the baselines with a small number of time steps, resulting in several times the speedup. However, annealing algorithms, both traditional methods and ours, require increased training overhead on larger datasets like ImageNet. We speculate that this is due to the need for more capacity [12, 22] and analyze this in our supplementary material.

6. Conclusion

This paper proposes RepAn, the first method to investigate the efficacy of re-parameterization (Rep) in enhancing model accuracy. While existing methods simply widen block structures by constructing diverse branches, our approach employs Rep as a training modality, utilizing a simple yet effective training paradigm that involves cycles of re-parameterization, expansion, restoration, and backpropagation operations. By capitalizing on Rep's lossless compression property, our method optimally activates the potential of the annealing algorithm. Moreover, RepAn is highly generalizable and compatible with all Rep structures, delivering performance improvements without incurring any extra inference-time costs.

Acknowledgement. This work was supported by National Science and Technology Major Project (No. 2022ZD0118202), the National Science Fund for Distinguished Young Scholars (No.62025603), the National Natural Science Foundation of China (No. U21B2037, No. U22B2051, No. 62176222, No. 62176223, No. 62176226, No. 62072386, No. 62072387, No. 62072389, No. 62002305 and No. 62272401), and the Natural Science Foundation of Fujian Province of China (No.2021J01002, No.2022J06001).

References

[1] RR Ade and PR Deshmukh. Methods for incremental learning: a survey. *International Journal of Data Mining & Knowledge Management Process*, 3(4):119, 2013. 4

[2] Ibrahim Alabdulmohsin, Hartmut Maennel, and Daniel Keysers. The impact of reinitialization on generalization in convolutional neural networks. *arXiv preprint arXiv:2109.00267*, 2021. 3

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48, 2009. 4

[4] Hamed Bonab and Fazli Can. Less is more: a comprehensive framework for the number of components of ensemble classifiers. *IEEE Transactions on Neural Networks and Learning Systems*, 30(9):2735–2745, 2019. 4, 5

[5] Hamed R Bonab and Fazli Can. A theoretical framework on the ideal number of classifiers for online ensembles in data streams. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2053–2056, 2016. 4, 5

[6] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 4

[7] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the twenty-first International Conference on Machine Learning*, page 18, 2004. 5

[8] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 8

[9] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018. 5

[10] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020. 8

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016. 7

[12] Yehuda Dar, Vidya Muthukumar, and Richard G Baraniuk. A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning. *arXiv preprint arXiv:2109.02355*, 2021. 8

[13] 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*, pages 248–255. Ieee, 2009. 2, 7

[14] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1911–1920, 2019. 1, 4, 6, 7

[15] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4510–4520, 2021. 3

[16] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Diverse branch block: Building a convolution as an inception-like unit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10886–10895, 2021. 1, 2, 4, 6, 7

[17] 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*, pages 13733–13742, 2021. 1, 2, 4, 5, 6, 7

[18] Xiaohan Ding, Honghao Chen, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Repmlpnet: Hierarchical vision mlp with re-parameterized locality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 578–587, 2022. 3

[19] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11963–11975, 2022. 3

[20] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16091–16101, 2023. 3

[21] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30: 681–694, 2020. 3

[22] Anna Golubeva, Behnam Neyshabur, and Guy Gur-Ari. Are wider nets better given the same number of parameters? *arXiv preprint arXiv:2010.14495*, 2020. 8

[23] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021. 4

[24] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *International Conference on Machine Learning*, pages 1311–1320. PMLR, 2017. 4

[25] Song Guo, Lei Zhang, Xiawu Zheng, Yan Wang, Yuchao Li, Fei Chao, Chenglin Wu, Shengchuan Zhang, and Rongrong Ji. Automatic network pruning via hilbert-schmidt independence criterion lasso under information bottleneck principle. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17458–17469, 2023. 3

[26] 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*, pages 770–778, 2016. 1, 2, 6, 7

[27] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017. 1

[28] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 4, 6

[29] Charles C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004. 5

[30] 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*, pages 1314–1324, 2019. 1

[31] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 7

[32] Mu Hu, Junyi Feng, Jiashen Hua, Baisheng Lai, Jianqiang Huang, Xiaojin Gong, and Xian-Sheng Hua. Online convolutional re-parameterization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 568–577, 2022. 7

[33] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017. 2, 5

[34] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017. 1

[35] Tao Huang, Shan You, Bohan Zhang, Yuxuan Du, Fei Wang, Chen Qian, and Chang Xu. Dyrep: Bootstrapping training with dynamic re-parameterization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 588–597, 2022. 3, 4, 6, 7

[36] Muhammad Hussain. Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*, 11(7):677, 2023. 1

[37] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015. 1, 2

[38] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 6

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[40] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 2

[41] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 7

[42] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017. 8

[43] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Autodeeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019. 1

[44] Ping Liu, Shizhong Han, Zibo Meng, and Yan Tong. Facial expression recognition via a boosted deep belief network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1805–1812, 2014. 4

[45] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2

[46] Gen Luo, Minglang Huang, Yiyi Zhou, Xiaoshuai Sun, Guannan Jiang, Zhiyu Wang, and Rongrong Ji. Towards efficient visual adaption via structural re-parameterization. *arXiv preprint arXiv:2302.08106*, 2023. 3

[47] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*, 2023. 3

[48] Sachin Mehta and Mohammad Rastegari. Mobilevit: lightweight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021. 3

[49] Kristine Monteith, James L Carroll, Kevin Seppi, and Tony Martinez. Turning bayesian model averaging into bayesian model combination. In *The 2011 International Joint Conference on Neural Networks*, pages 2657–2663. IEEE, 2011. 5

[50] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 4, 5

[51] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(4):497–508, 2001. 4

[52] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1

[53] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951. 7

[54] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018. 4

[55] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016. 4

[56] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 6, 7

[57] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated Annealing: Theory and Applications*, pages 7–15. Springer, 1987. 2

[58] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. An improved one millisecond mobile backbone. *arXiv preprint arXiv:2206.04040*, 2022. 1, 3, 7, 8

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017. 3

[60] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023. 3

[61] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6427, 2021. 4

[62] Jingjing Xie, Bing Xu, and Zhang Chuang. Horizontal and vertical ensemble with deep representation for classification. *arXiv preprint arXiv:1306.2759*, 2013. 5

[63] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1492–1500, 2017. 7

[64] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*, pages 702–717. Springer, 2020. 4

[65] Sergey Zagoruyko and Nikos Komodakis. Diracnets: Training very deep neural networks without skip-connections. *arXiv preprint arXiv:1706.00388*, 2017. 1

[66] Mingyang Zhang, Xinyi Yu, Jingtao Rong, Linlin Ou, and Feng Gao. Repnas: Searching for efficient re-parameterizing blocks. *arXiv preprint arXiv:2109.03508*, 2021. 3, 7

[67] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890, 2017. 8

[68] Xiawu Zheng, Xiang Fei, Lei Zhang, Chenglin Wu, Fei Chao, Jianzhuang Liu, Wei Zeng, Yonghong Tian, and Rongrong Ji. Neural architecture search with representation mutual information. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11912–11921, 2022. 3

[69] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 3