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

1. Analyses of RepAn

In this section, we further analyze the gradient during training through the impact of the loss function, and discuss the limitations of training consumption on ImageNet [1].

1.1. Gradient Analysis: More Details

Let $\boldsymbol{X} \in \mathbb{R}^{C_{in}\times H\times W}$ and $\boldsymbol{Y} \in \mathbb{R}^{C_{out}\times H'\times W'}$ represent the input and output feature maps, respectively. The forward propagation through a convolutional layer is formulated as $\boldsymbol{Y} = \boldsymbol{X} \circledast \boldsymbol{W} + \boldsymbol{b}$, where $\boldsymbol{W} \in \mathbb{R}^{C_{out}\times C_{in}\times K\times K}$ and $\boldsymbol{b} \in \mathbb{R}^{C_{out}}$ are the weights and bias terms of the convolution, respectively.

For the expanded branches used in our method, their gradient is calculated as:

$$\begin{cases} \frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{W}_{\text{exp}}} = \boldsymbol{X} \circledast \lambda \sum_i \left(\frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{Y}} \cdot \frac{\gamma^{(i)}_{\text{exp}}}{\sigma^{(i)}_{\text{exp}}} \right), \\ \frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{b}_{\text{exp}}} = \lambda \sum_i \left[\sum_{u,v} \left(\frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{Y}} \right)_{u,v} \cdot \frac{\gamma^{(i)}_{\text{exp}}}{\sigma^{(i)}_{\text{exp}}} \right], \end{cases} \tag{1}$$

where C is the other term which is not related to the convolution parameters $\boldsymbol{W}$ and $\boldsymbol{b}$, and λ is the adjustment scheme $attach\_rate$, as noted in Sec. 3.3 in our paper. Note that the expanded branches have a larger gradient magnitude at the beginning of training. We interpret this phenomenon as inheriting prior knowledge and achieving weights ensemble. The term $\frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{Y}}$ in Eq. (1) represents the ratio of the loss function to the output, which is directly related to the obtained gradient value of the convolution parameters.

Furthermore, since we use the cross entropy error as the loss criterion $f(\boldsymbol{Y})$, we have:

$$\frac{\partial f(\boldsymbol{Y})}{\partial y_t} = -\frac{\partial \sum_{t=0}^{n} y_t \log(\hat{y}_t)}{\partial \hat{y}_t} = -\frac{y_t}{\hat{y}_t}, \tag{2}$$

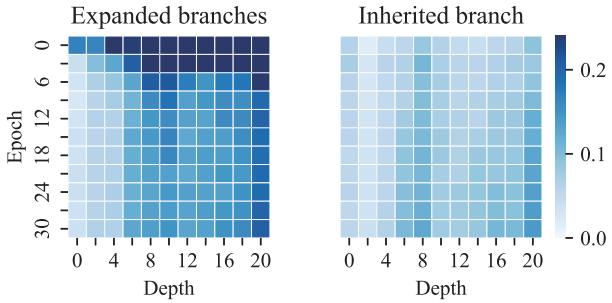

Figure 1. Comparison of gradient values of different depths and training epochs for expanded and inherited branches.

where $\hat{y}$ denotes the predicted distribution. Then, the gradient is calculated as follows:

$$\begin{aligned} \frac{\partial f(\boldsymbol{Y})}{\partial x_i} &= \frac{\partial f(\boldsymbol{Y})}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial x_i} + \sum_{t=1,t\neq i}^{n} \frac{\partial f(\boldsymbol{Y})}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial x_i} \\ &= \frac{\partial f(\boldsymbol{Y})}{\partial \hat{y}_i} \cdot (\hat{y}_i - \hat{y}_t \cdot \hat{y}_i) + \sum_{t=1,t\neq i}^{n} \frac{\partial f(\boldsymbol{Y})}{\partial \hat{y}_t} \cdot (-\hat{y}_t \cdot \hat{y}_i) \\ &= -\frac{y_t}{\hat{y}_t} \cdot (\hat{y}_i - \hat{y}_t \cdot \hat{y}_i) + \sum_{t=1,t\neq i}^{n} -\frac{y_t}{\hat{y}_t} \cdot (-\hat{y}_t \cdot \hat{y}_i) \\ &= \hat{y}_i \cdot y_i - y_i + \sum_{t=1,t\neq i}^{n} \hat{y}_i \cdot y_t \\ &= -y_i + \sum_{t=1}^{n} \hat{y}_i \cdot y_t \\ &= -y_i + \hat{y}_i. \end{aligned} \tag{3}$$

By summing Eq. (3), we have:

$$\frac{\partial f(\boldsymbol{Y})}{\partial \boldsymbol{X}} = \hat{\boldsymbol{Y}} - \boldsymbol{Y}. \tag{4}$$

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

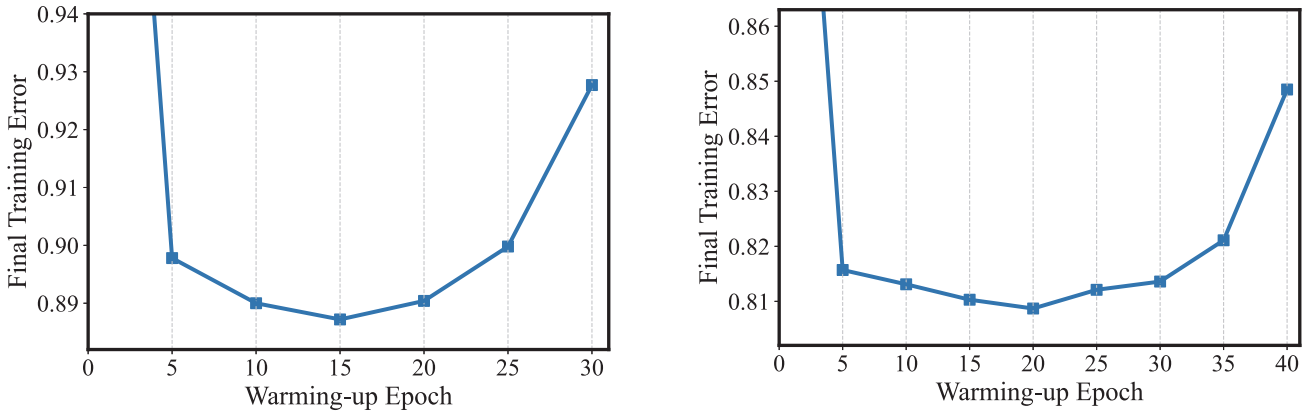

Figure 2. Comparison of choosing different warm-up epochs for λ. **Left**: Training for 30 epochs each time step. **Right**: Training for 40 epochs each time step.

This formula shows that when the output predicted by the network is close to the real labels, it is less affected by back-propagation during training. Figure 1 compares the gradient values during training, and the experimental results verify this precisely. Specifically, the gradient values of the extended branches are generally greater than that of the original branch that inherits the previous weights. As the original branch utilizes the lossless compression property of Rep, it **fully preserves the output of the previous stage**, being less modified during training.

From the perspective of the training epoch, it can be found that the expanded branches have greater gradients in the early period of training. This is because the network has just been randomly initialized and thus has a **rapid learning process in the early stages**. Considering that this introduces significant oscillations to the training process, we thus present the parameter λ to reduce the gradient value to stabilize the training artificially.

From the perspective of network depth, deeper layers always have greater gradient values. We understand this process as a **feature fusion process** of multiple branches. Therefore, more learning is required in the layers close to the output, leading to improved gradient values of the deeper layers.

1.2. Discussion: Training on ImageNet

Our paper mentions that RepAn needs more learning epochs when applied to larger datasets. Specifically, setting the number of epochs of each time step to 30 is sufficient for the CIFAR-10/100 datasets [5], and we run for up to 5 time steps for better convergence. This process makes the total number of training epochs even lower than regular training, achieving a speedup.

However, it is hard to achieve remarkable training cost savings on large datasets and heavyweight neural networks. On the one hand, more epochs are required for the significantly increased amount of training data (*e.g.*, 1.28M for ImageNet and 60K for CIFAR) on larger datasets. On the other hand, the number of parameters of the network also multiplies, resulting in a much slower learning process.

Moreover, the cyclic cosine annealing method requires multiple training time steps. Annealing methods [4, 6] achieves better final performance through a larger learning rate, fewer learning epochs, and multiple times of training to convergence. Our training paradigm matches this annealing process and thus exhibits similar properties. Using a larger learning rate and reducing the number of training epochs may allow for better annealing. Therefore, we compare the impact of different training epochs in Sec. 2.2, and find that insufficient training will seriously affect the convergence at each time step, resulting in further degradation of performance. Although RepAn can bring remarkable performance improvements, the training overhead in large datasets cannot be significantly reduced, which can be further improved in future work.

2. Additional Ablation Studies

This section introduces some additional ablation experiments mentioned in the original paper, and detailed analyses of the experimental results.

2.1. Influence of Schedulers for λ

The adjustment scheme $attach\_rate$, represented as λ, is designed to achieve stability in the early stage of training and reduce the oscillation caused by random initialization of extra branches. In the later stages of training, the value of λ should be restored to 1.0 to avoid affecting the actual process of training. The adjustment of λ is similar to the warm-up process and allows artificially adjusting its value at different epochs. A simple linear warm-up scheduler is

designed as:

$$\lambda \leftarrow \max\left(\frac{\#\text{train\_epochs}+1}{\#\text{warmup\_epochs}+1},\ 1.0\right). \qquad (5)$$

This scheduling process can also be designed as a more complex function, but a linear approach is sufficient due to the small training epochs involved. We then compare the effect of different warm-up epochs using the RepVGG [2] network on CIFAR-100, as shown in Fig. 2.

When λ is not used, the network converges with a significantly larger loss, resulting in poor performance. When a suitable number of warm-up epochs is set, the network can obtain better convergence. Using less than half of the total epochs helps achieve optimal performance. However, too many warm-up epochs reduce the final performance, but **still perform better than not using** λ.

However, this threshold may vary depending on different datasets and the capacity required by the network. In practice, we reduced the number of warm-up epochs of heavy-weight models on ImageNet, leaving more epochs for better training to convergence.

2.2. Influence of Training Epochs on ImageNet

As mentioned in Sec. 1.2, training on ImageNet requires more epochs to meet the learning requirements for larger capacity and more parameters. We compare different training epochs, aiming to find the minimum cost that satisfies the training of the annealing algorithm. We use ResNet [3] for comparison, following the settings in our main experiments but changing the training epochs only.

Figure 3 shows the training curves of different training epochs. When insufficient training epochs are used, the network cannot be fully trained, resulting in a poor final performance. This figure shows that using 80 training epochs is enough for a satisfying convergence, but more epochs always performs slightly better.

Although using fewer single training epochs decreases the training performance, due to the cyclic cosine annealing algorithm [6], the network benefits from the multiple convergences to discover a better optima, thus performs better [4]. Experimental results in our paper also demonstrate the effectiveness of this training method.

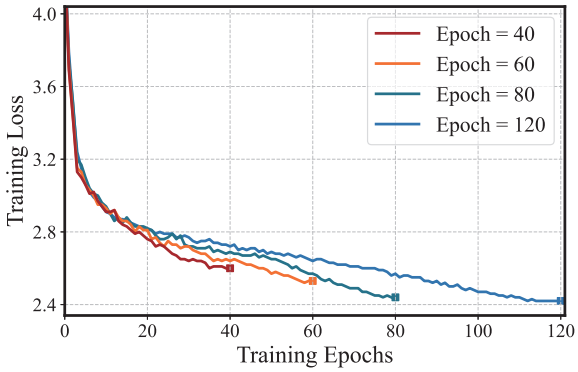

Figure 3. Comparison of different number of epochs for training ResNets on ImageNet. Best viewed in color.

References

[1] 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. 1

[2] 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. 3

[3] 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. 3

[4] 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, 3

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

[6] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2, 3